MULTI-OBJECTIVE OPTIMAL DESIGN OF STEEL TRUSSES IN UNSTRUCTURED DESIGN DOMAINS

A Thesis

by

SANGWOOK PAIK

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2005

Major Subject: Civil Engineering

MULTI-OBJECTIVE OPTIMAL DESIGN OF STEEL TRUSSES IN UNSTRUCTURED DESIGN DOMAINS

A Thesis

by

SANGWOOK PAIK

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Anne M. Raich
Committee Members,	Mary Beth Hueste
	V. Jorge Leon
Head of Department,	David Rosowsky

August 2005

Major Subject: Civil Engineering

ABSTRACT

Multi-Objective Optimal Design of Steel Trusses in Unstructured Design Domains.

(August 2005)

Sangwook Paik, B.S., In-ha University, South Korea

Chair of Advisory Committee: Dr. Anne M. Raich

Researchers have applied genetic algorithms (GAs) and other heuristic optimization methods to perform truss optimization in recent years. Although a substantial amount of research has been performed on the optimization of truss member sizes, nodal coordinates, and member connections, research that seeks to simultaneously optimize the topology, geometry, and member sizes of trusses is still uncommon. In addition, most of the previous research is focused on the problem domains that are limited to a structured domain, which is defined by a fixed number of nodes, members, load locations, and load magnitudes.

The objective of this research is to develop a computational method that can design efficient roof truss systems. This method provides an engineer with a set of near-optimal trusses for a specific unstructured problem domain. The unstructured domain only prescribes the magnitude of loading and the support locations. No other structural information concerning the number or locations of nodes and the connectivity of members is defined. An implicit redundant representation (IRR) GA (Raich 1999) is used in this research to evolve a diverse set of near-optimal truss designs within the specified domain that have varying topology, geometry, and sizes. IRR GA allows a

Pareto-optimal set to be identified within a single trial. These truss designs reflect the tradeoffs that occur between the multiple objectives optimized.

Finally, the obtained Pareto-optimal curve will be used to provide design engineers with a range of highly fit conceptual designs from which they can select their final design. The quality of the designs obtained by the proposed multi-objective IRR GA method will be evaluated by comparing the trusses evolved with trusses that were optimized using local perturbation methods and by trusses designed by engineers using a trial and error approach. The results presented show that the method developed is very effective in simultaneously optimizing the topology, geometry, and size of trusses for multiple objectives.

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge my advisor, Dr. Anne M. Raich, for her guidance and support throughout my graduate studies and the enormous effort she made to revise this document. I also wish to sincerely acknowledge the contribution of Dr. Mary Beth Hueste, Dr. Jorge V. Leon and Dr. Jose Roesset for their guidance and helpful review of this document.

I would like to thank my parents and sisters for their love and encouragement all the time. Finally, thanks be to God for his everlasting love.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Page

Page

Page

Page

Page

LIST OF TABLES

Page

1 INTRODUCTION TO RESEARCH ON MULTI-OBJECTIVE DESIGN OF STEEL TRUSSES

1.1 Introduction

Modern structural systems have become more complicated to design and construct due to increasing span lengths and loads and also due to specific architectural requirements on space and aesthetics. Material and construction costs have also continued to increase. Therefore, the efficiency of the structural design selected must be considered by the design engineer. Being able to significantly enhance the efficiency of a design depends on a designer's experience and is at heart a trial-error procedure. In order to assist the engineer in this process, advancing research in the area of structural design optimization is necessary. One type of structural system that can be benefit greatly from computer-based assistance is the optimization of long-span trusses because, in this case, cost-efficient design is directly related to the reduction of construction costs and the satisfaction of the designer's aesthetic criteria.

1.2 Motivation

The importance of cost-efficient design of trusses has triggered an increase in research in the field of global optimization methods. Mathematical methods, which have inflexible constraints in the formulations and perform a local search, are not suitable for

---

This thesis follows the style and format of *Journal of Structural Engineering.*

the design problem domain. Instead, genetic algorithms (GAs) have been adapted widely for design optimization along with other heuristic methods like simulated annealing and taboo search. These methods have been proven to be robust in discrete variable domains. From sizing optimization to optimization of unstructured problem domains, there has been substantial research performed on truss optimization through the development of various advanced GAs. Many researchers have worked on problems that are limited to a defined structured domain, which has fixed number of nodes, members and load locations. Research that seeks to simultaneously optimize topology, geometry, and size of trusses, however, is uncommon. There has also been less research performed concerning multi-objective optimization of truss topology and geometry.

Even though some previous research investigated performing design optimization in an unstructured problem formulation, the optimization algorithms used were not flexible enough to explore widely the unstructured domains due to many constraints or the inflexibility of the design representation used by simple GA. In order to perform optimization in an unstructured domain, the development of an advanced representation and computational method that enables various truss topologies and geometries to be generated and synthesized simultaneously is inevitable. In this research, a computational method that uses an implicit redundant representation GA (IRR GA) is developed to simultaneously generate and synthesize of a diverse range of truss designs in order to explore the unstructured design domain. In this research, a set of near-optimal alternative truss designs is obtained instead of a single optimum design. The set provides the design engineer with information concerning tradeoffs that occur between

the objectives being optimized.

1.3 Objective and Scope

The objective of this research is to develop a computational method that can design efficient roof trusses. The method is capable of performing topology, geometry, and size optimization simultaneously. Using this method, a set of near-optimal trusses for a defined unstructured problem domain can be obtained. The unstructured domain approaches the conceptual design problem formulated by only prescribing the magnitude of loading and the support locations. No other structural information concerning nodal locations or the number or the placement of members is defined. The IRR GA is used in this research to evolve a diverse set of near-optimal trusses within this domain that have varying topology, geometry, and member sizes.

The trusses designed will be optimized to meet three objectives that concern stress, weight, and deflection. A Pareto-optimal curve (weight and deflection) and a Pareto-optimal surface (weight, deflection, and stress), which represents the optimal set of trusses, will be generated for each predefined unstructured problem domain (loading condition and support locations that are defined by the span length). The Pareto-optimal curve obtained will be used to provide design engineers with a set of efficient conceptual designs from which they can select their final design. The quality of the Pareto-optimal set of designs obtained for each problem domain will be evaluated by comparing the trusses evolved with trusses optimized in previous research efforts and with trusses designed based on standard engineering practice.

1.4 Methodology

The following design and optimization issues were investigated to achieve the objective of this research:

1. *Design an IRR GA parameter value representation for a truss structure.*

   The encoding language used by a GA can strongly influence its performance. In the truss design domain, the size of the feasible design area with respect to the infeasible design area fully depends on the representation used by GA. To generate diverse alternative designs and assist in synthesizing designs, a flexible encoding language that represents a wide variety of topologies, geometries, and shapes of trusses was designed.

2. *Modeling the unstructured domain in the truss problem*

   In an unstructured domain, no information of the truss structure is explicitly provided. To evolve the structures, however, the environment has to be prescribed. In this research, the distributed load, span length, maximum height of structures, and a set of ten steel cross-section shapes were predefined as the environment used to evolve designs.

3. *Define and investigate the effect of the form of the fitness function, parameters and string length*

   To explore a large portion of the feasible design domain, which allows the discovery of more efficient design alternatives, the individuals in a population should not converge to a local optimum too quickly. To obtain near-optimal solutions, it is also important to balance the population

during exploration between the feasible and infeasible area. For effective exploration and balance of feasible and infeasible individuals, a set of optimum parameters (crossover probability and mutation probability) should be defined. In addition for the IRR GA, the proportion of feasible/infeasible area, the design complexity, diversity, and the redundancy of the GA provided depends on the string length used to encode individuals. In this research, an efficient parameter value set and string length were determined using a composite fitness function.

4. *Develop the strategies and architecture of the IRR GA for multi-objective optimization*

   The modified MOGA developed focused on solving two problems. The first was to obtain design alternatives, which is the objective of this research, and the other was to prevent the premature removal of individuals from a population before they have a chance to improve and compete. In order to meet these goals, some advanced MOGA concepts were adapted and modified and a new architecture for the computational method was developed.

5. *Verify and compare the results obtained from the modified MOGA.*

   First, the optimal truss alternatives generated from this research were locally refined to verify the degree of sizing optimization that was obtained by the modified MOGA. Then, the trusses obtained using the MOGA were compared with trusses that were designed by trial-error and

by local optimization methods.

1.5 Outline

Section 1 introduction presents a brief discussion of the motivation, objective, scope, and methodology for this research. In Section 2, an overview is presented concerning previous research in the area of truss optimization. The basic formulation and operators for a simple genetic algorithm (SGA) and implicit redundant representation GA are also introduced. In Section 3, the unstructured design problem domain and the IRR GA representation used for trusses in this research are presented. Section 4 presents the procedures of adjustment by experiment required to obtain efficient parameters and string lengths based on the representation and the unstructured problem domain defined previously. In Section 5, the modified MOGA that is proposed in this research is presented. In addition, other MOGA strategies and concepts that were adapted in this research are introduced. Section 6 presents the verification and the comparison of the results of this research in order to evaluate the performance of the method developed. The conclusions, including a discussion of the problems identified by this research and future recommendations, are presented in Section 7.

2 LITERATURE REVIEW

2.1 Introduction

This Section consists of three parts. In the first part, the format and operators used for simple GAs are briefly discussed. The second part provides a summary of previous research on truss optimization using GAs. The summary focuses on the methodology used and the strategies required in each research effort. The advanced representation GA used in this research, which is called the implicit redundant representation (IRR) (Raich and Ghaboussi 1998), is introduced and discussed in the last part of this Section.

2.2 Overview of Genetic Algorithms

2.2.1 Introduction

Mathematical search methods, which have been studied substantially in the past, can be classified into one of two methods. One is as an indirect method, in which the local optima are obtained by setting the gradient of the objective function to zero. The other is as a direct method, in which the search for local optima progresses moving toward the solution using local gradient information. In both methods, the solution is based on a local scope and the final optima found heavily depend on the initial starting point and on the neighborhood investigated during search. In reality, however, most problems have unpredictable, complex domains, in which continuity and the existence of gradient information are not guaranteed. For these reasons, mathematical methods

developed for optimization do not have the robustness required to search structural design domains that tend to have non-smooth (discontinuous) and highly nonlinear domains with small yet complex feasible search spaces.

The genetic algorithm was first proposed by Holland (1975) and further developed by Goldberg (1989). Genetic algorithms (GAs) are search methods modeled after natural genetic systems. These systems have the characteristic that they evolve through combination and mutation to adapt themselves to their environments. Many researchers have validated the performance of GAs on optimization problems. One benefit of GAs is that they are easy to implement computationally,while still providing powerful search methods. In addition unlike traditional search methods (mathematical methods), GAs are robust and can search discontinuous problem domains and are not limited by restrictive assumptions of search domain. Goldberg (1989) identified the four main differences of GAs from other traditional optimization methods:

1. GAs work with a coding of the parameters set, not the parameters themselves
2. GAs search from a population of points, not a single point.
3. GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.
4. GAs use probabilistic transition rules, not deterministic rules.

2.2.2 Simple Genetic Algorithm (SGA)

Fig. 2.1 presents the flowchart defining a SGA. There are three main operations: selection, crossover, and mutation. Each parameter value is represented as n-bit binary number. The initial population consists of the individuals that are randomly created. To evaluate each individual in each generation, the binary numbers are decoded into parameter values by a rule, which is predetermined by a mapping of the parameter values to the encoded binary values. The procedure is repeated until improvement is not found, or pre-specified generation is completed. De Jong (1975) conducted research to determine the performances of GAs that implemented simple crossover, simple mutation,

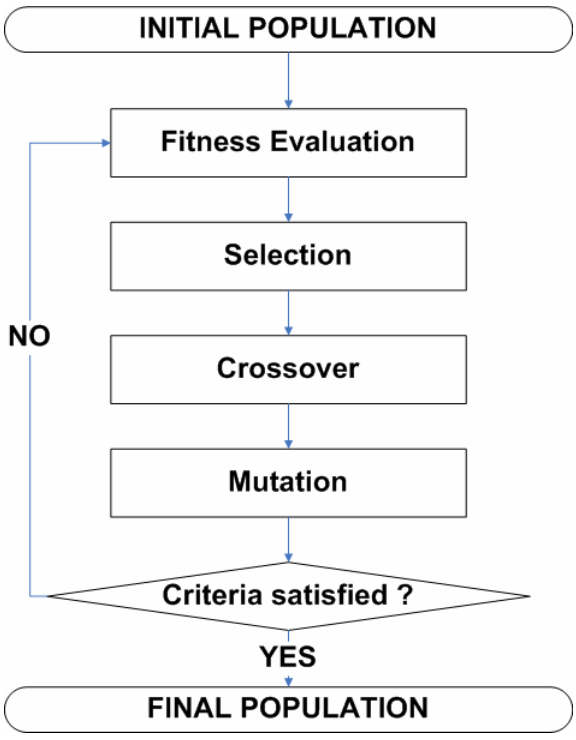


Fig. 2.1. Flow of simple genetic algorithm

and roulette wheel selection using five test functions. This GA has been called the simple genetic algorithm (SGA). In the last two decades, SGA has been used for a wide variety of applications along with different combinations of GA operators, parameter settings, and search strategies.

2.2.2.1 SGA Representation

All optimization design variables are represented by binary numbers in a SGA individual. As shown in Fig. 2.2, two design variables are converted into binary numbers, in which each variable is encoded using 4 binary bits. The range of each variable in this case is $0 \sim (2^4\text{-}1)$. An individual is then defined by concatenating these binary numbers together to form a string. Each individual in the initial SGA

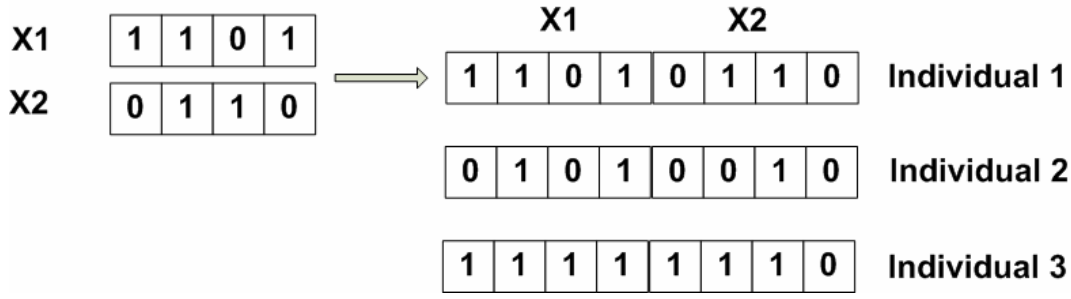


Fig. 2.2. Example of the representation of SGA

population is randomly initialized. During optimization, each individual in the current population is decoded (converted into decimal numbers) by SGA and the decoded values are used to evaluate the fitness of each individual. The fitness of each individual

is determined using a fitness function. Typically, the fitness function measures how well the individual meets the stated objectives along with any penalties on fitness for constraint violations. For instance, a SGA can be applied to an optimization problem with a single objective function:

$$\text{Max } F(\mathbf{x}) = \sum_{i=1}^{m} \mathbf{x}_i^2 \qquad (2.1)$$

The fitness value of each individual in the SGA population is then computed using equation (2.1). Table 2.1 shows the fitness values of each individual.

Table 2.1. Fitness value of the individuals computed with equation 2.1 and values in Fig. 2.2

Individual	x_1	x_2	Fitness ($x_1^2+x_2^2$)
1	13	10	269
2	5	2	28
3	15	14	421

2.2.2.2 Selection Operator

Based on their fitness values, individuals are selected for the next generation. Fitter individuals have a higher probability for selection than those with lower fitness. The selected individuals are used to generate new offspring through crossover and mutation to construct the population for the next generation. One selection scheme that is commonly used is roulette wheel selection, which is also called fitness proportional selection. In this type of selection, individuals in the population have a probability of selection in proportion to their fitness:

$$\rho(x_i) = \frac{F(x_i)}{\sum_{i=1}^{m} F(x_i)} \tag{2.2}$$

Another selection scheme that is often preferred is tournament selection. In this type of selection, a predefined number of individuals (tournament set) are drawn randomly, and only the one with the highest fitness out of the tournament set is selected. This process is continued until the next generation population is full. Fig. 2.3 shows an example of tournament selection. Individuals that have higher fitness have a greater chance of being selected, which results in generating fitter individuals in next generation. Tournament selection allows the user to have greater control over the selection pressure from one generation to another. A smaller tournament size will reduce the selection pressure. Having to high of a selection pressure leads to premature convergence of the population, which results typically in only obtaining a local optimum.

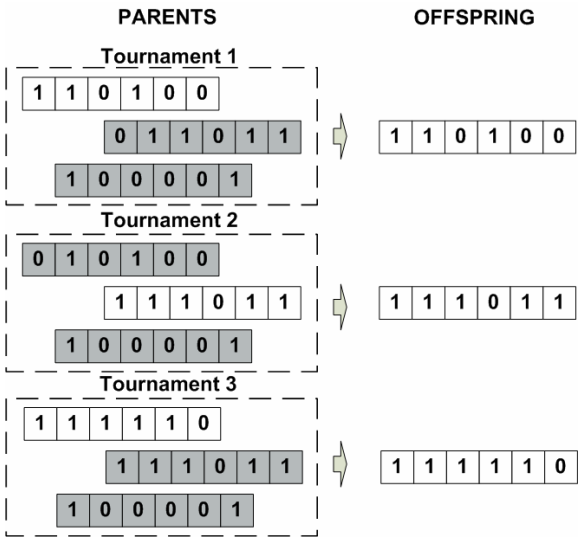


Fig. 2.3. Example of tournament selection (tournament size: 3)

2.2.2.3 Genetic Operators in SGA

The selected individuals are manipulated by application of crossover and mutation operators. Crossover is a recombination operator. Therefore, crossover cannot create new information. By recombination and reposition of the already exist information, however, crossover can provide a greater possibility for individuals to be improved. In the SGA, information encoded in a pair of individuals is swapped based on a single random crossover point as shown in Fig. 2.4. Mutation helps to prevent the population from losing diversity and to introduce or reintroduce information into the population. Supporting diversity, mutation enables individuals in population to continue to explore the search.

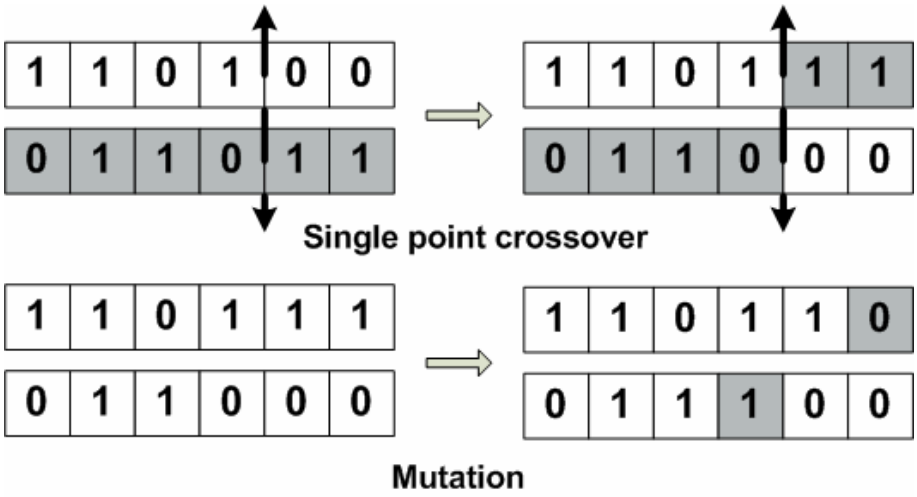


Fig. 2.4. Application of single point crossover and mutation for a pair of selected individuals

2.2.3 Elitism

The stochastic selection process performed by GAs cannot guarantee the selection of any specific individuals, which means that even an individual that has the highest fitness can possibly be removed from the population. Elitism preserves the highest fitness individuals without any manipulation through to the next generation (exploitation), which may result in the GA process may converging faster to a near-optimal solution. Elitism, however, may decrease the chance that individuals in the population explore more of search domain, which causes individuals to converge to a local solution rather than a global solution. Therefore, the balance between the exploitation of solutions in the current population and the exploration of search space for new solutions should be considered for each problem domain.

2.3 Overview of Previous Research on Truss Design Optimization Using GA

2.3.1 Non-Smooth Problem Domain

Many calculus-based methods are efficient and accurate in searching small and smooth search domains. However, structural design problems have often very complicated and non-smooth search domains. GAs are robust and efficient enough in non-smooth and complicated problem domain, as has been shown by the results obtained by researchers. One early research effort was the sizing optimization of a truss structure, in which the member sizes of the truss were optimized for a fixed topology and geometry. Based on this research conducted by Rajeev and Krishnamoorthy (1992), the effectiveness of a SGA was shown for a structural design problem. In the research,

discrete sizing optimization was performed to optimize three-bar, 10-bar, and 25-bar trusses. A set of ten cross-sectional areas were defined to select from to optimize the weight. The results showed that SGA found a better minimum weight than other traditional optimization methods.

2.3.2 The Sizing, Shape, and Topology Optimization Using Ground Structure

In order to enhance the capability of design optimization, truss topology configuration and shape optimization were performed by many researchers in addition to sizing optimization. In these research studies, a ground structure approach, in which all possible connections to given nodes are predefined as shown Fig. 2.5, has often been adopted. Using the ground structure approach, which was first proposed by Dorn et al. (1964), topology optimization was performed by Hajela and Lee (1995). In this case, the structural problem domain was defined by a number of nodes and every node was connected to each other as shown in Fig. 2.5. The presence and absence of each element in the truss design was determined by the values decoded from the GA genes. In the research, a number of truss topologies, which were kinematically stable, were generated without the consideration of the other structural constraints. In the next stage using the topologies generated from the first step as initial seeds, sizing optimization was performed with the consideration of structural constraints in addition to additional topology optimization.

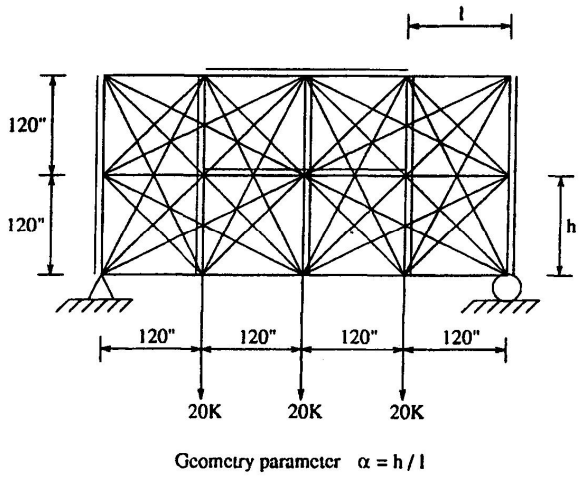


Fig. 2.5. Bridge type ground structure (Hajela and Lee 1995)

Using the ground structure, sizing optimization along with limited geometry and topology optimization were performed by Rajan (1995). Previous research efforts usually used multi-step processes. In this optimization process, sizing design variables (the cross-section areas of the members), topology design variables (DOF at specified joint and the presence or absence of members), and geometry design variables (y-coordinates of specified joints) were simultaneously considered. Using restarts with new seeds that were the results from the previous process, the computational expense was reduced. Although the research showed an improvement compared to previous research concerning the optimization of truss designs, the research still had the drawbacks of the ground structure approach, in which all the information of possible topologies should be explicitly encoded in each individual. Therefore, the enough flexibility for creating diverse designs in an unstructured design problem domain could not be provided.

In the research by Deb and Gulati (2001) that did not use a ground structure approach, each member's presence or absence in the truss topology was determined by the area assigned to each member of a truss. If the member size was less than a predefined critical area then the member was removed. In the research, the nodes of a truss were divided into two classes. One class consisted of the basic nodes, which were nodes required to make a truss that was stable and able to carry loads. The other class consisted of the non-basic nodes, which were optional nodes that could be removed. By defining the two classes, the trusses that did not have a basic node were excluded from further evaluation.

2.3.3 Adaptive GAs and Operators

As previously mentioned, GAs have been shown to be robust and efficient in searching a design problem domain by a number of researchers. However, there exist many problems that must be overcome before these methods can be applied to conceptual design and optimization in complex design problems. Issues include the flexibility of the GA representation for an unstructured domain, the computational expense, and the process time and performance in obtaining near-optimal solutions. Many researchers have proposed various strategies and operators to address these problems. This research effort focuses on providing a flexible representation to increase the performance and obtain quality near-optimal solutions.

2.3.3.1 Variable-Complexity Genetic Algorithm

A standard GA has a fixed string length. This causes a severe limitation on the representation in that all possible design variables must be explicitly encoded in each GA individual. A modified crossover operator was proposed by Gage, Kroo, and Sobieski (1995), in which the selected parents had different crossover points and consequently the created offspring had different string lengths (i.e. different topologies). Fig. 2.6 illustrates the crossover operation based on different crossover points. The Offspring 1 and Offspring 2 created by crossover using Parent 1 and Parent 2 have different string lengths, which are decoded into different topologies. This variable-complexity genetic algorithm enabled GAs to only encode the necessary genes that defined different design topologies. This avoided the problem with the ground structure

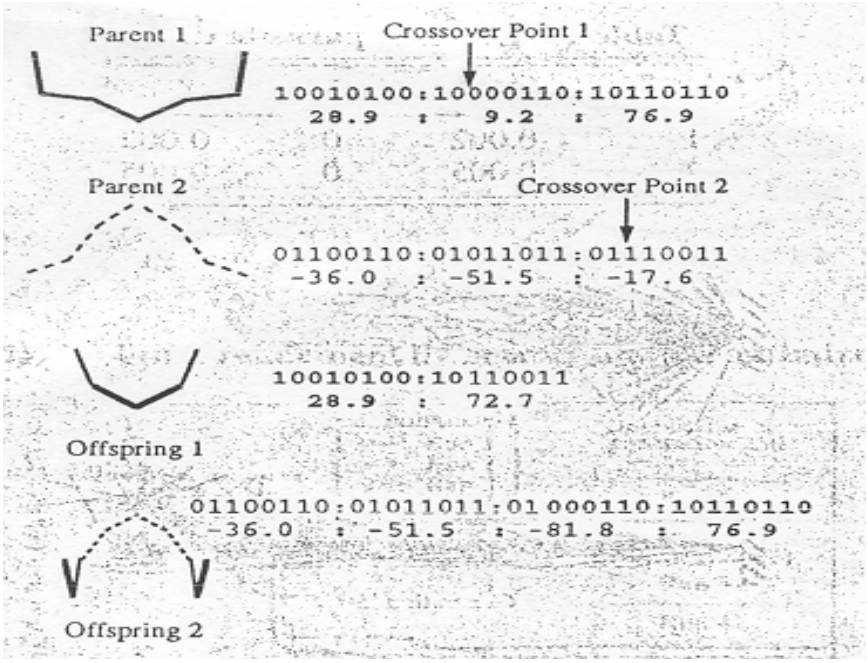


Fig. 2.6. Modified crossover operator (Gage, Kroo, and Sobieski 1995)

approach, in which all the information for all possible members and nodes must be encoded. This approach allowed GAs to optimize the topology of design, provided the SGA with some flexibility, and showed the possibility of performing optimization in a fully unstructured domain problem.

2.3.3.2 Variable String Length Genetic Algorithms (VGA)

Simple genetic algorithms cannot be directly used for topology optimization, in which the number of nodes, members, and their connectivity changes. To overcome the problem, previous researchers used a ground structure approach to perform limited topology optimization. However, the SGA using ground structures does not scale well with problem size which impacts the quality of the solution obtained because every member in the ground structure must be encoded to indicate the absence or presence of a member. To overcome this limitation of SGA, VGA (variable length genetic algorithm) was proposed by Rajeev and Krishnamoorthy (1997). Unlike SGA, strings with various lengths, which are determined by control variables, exist in a same population. Through crossover, string lengths are swapped or exchanged. In the research conducted by Rajeev and Krishnamoorthy (1997), VGA was applied to 10 bar, seven bar, six bar, and four bar trusses. Over the GA iterations performed, individuals in the population tended toward encoding the same string length, which showed a lack of diversity and flexibility. In another trial using a bracing pattern, topology optimization was performed without variation of string length. In this problem, the number of bracing panels was determined as the control variable, and the same length of string, which contained

structural configurations, was assigned to each panel. By using the bracing pattern, various shapes of trusses were able to be encoded using the same string length.

2.3.3.3 Optimization in an Unstructured Problem Domain

In the research conducted by Shrestha and Ghaboussi (1998), a new methodology to evolve a optimum truss design in an unstructured design domain was proposed. Each individual consisted of several substrings that contained information corresponding to each node. Each substring encoded the nodal location (x-coordinate, y-coordinate), the presence/absence code of a node, and the member information (section size, the presence/absence code of each member, priority) that were connected to the node as shown in Fig. 2.7. Even though a fixed string length was used, the presence/absence value in the substrings enabled the structures in the population to have a various number of elements and nodes. To investigate the potential of this methodology, optimization trials were performed for two design spaces having maximum heights of 10 m, and 35 m with both spanning 70 m). The truss weight was minimized with several constraints. The two trusses generated for the two design spaces had reasonable configurations and the stresses in each member were within the conditions stated by the prescribed loading and design space. This research result showed the adaptability of the SGA to unstructured problem domains.

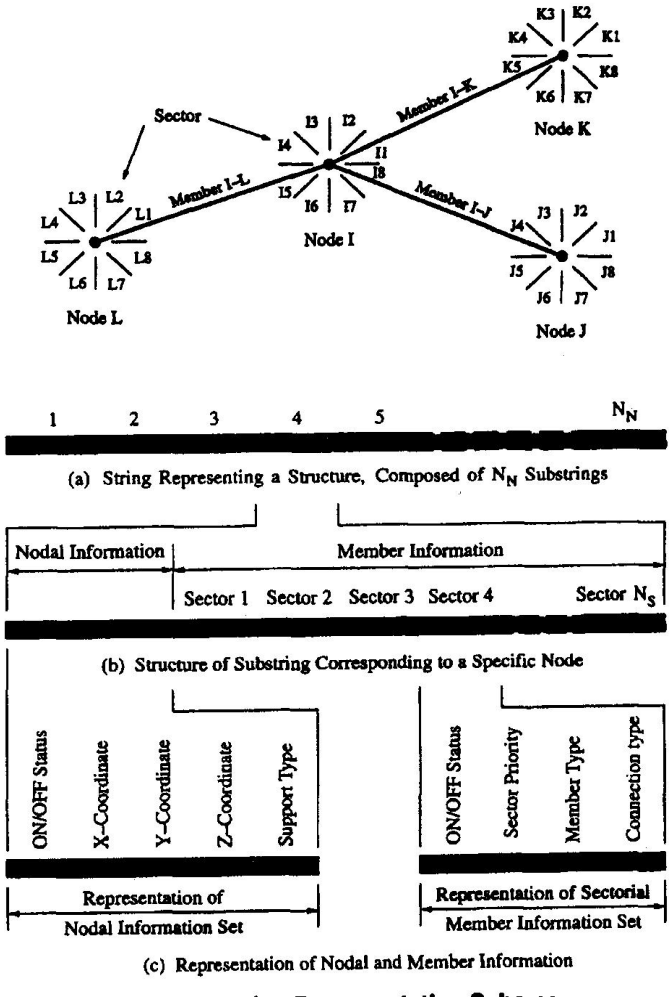


Fig. 2.7. Representation scheme for members (top) and representation format (bottom) (Shrestha and Ghaboussi 1998)

2.3.3.4 Crossover and Mutation for Variable String Lengths

Allowing different topologies to exist in a single population requires different string lengths, Therefore, crossover can be classified as two operators (inter-crossover and infra-crossover) to handle variable string lengths while the mutation operator remains similar to that used in traditional GAs (Ryoo and Hajela 2004). The infra-

crossover is allowed only if the two strings selected have the same length. The inter-crossover is operated as a traditional GA crossover, which means that even if the two string lengths are different, crossover is performed. Fig. 2.8 illustrates inter-crossover, in which the crossover sites are selected to fall within the shorter string. The selected substrings are then swapped between the two strings.

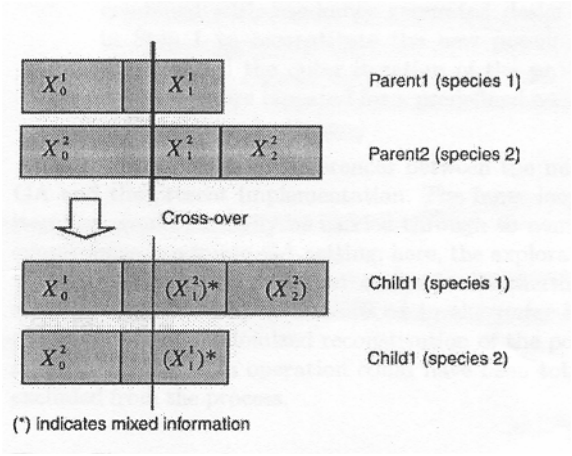


Fig. 2.8. Inter-species crossover (Ryoo and Hajela 2004)

Ryoo and Hajela (2004) had adapted the micro-GA, which was first proposed by Krishnakumar (1989). In the micro-GA, the population size is smaller (about 2~4 individuals) than traditional GA, therefore the individuals converge very rapidly to a single, often non-optimal solution. Ryoo and Hajela (2004) combined inter-crossover and infra-crossover. The GA process was formed with two loops (Inner Loop and Outer Loop). In the inner loop, the infra-crossover was operated to find local optimum solutions between the individuals that have the same string length (i.e. same topology).

And in the outer loop, the inter-crossover was operated between the locally optimized individuals and new individuals that were generated randomly, which enabled the population to retain diversity. The research results of the adaptive micro-GA resulted in more optimized design as well as faster convergence than traditional GAs with only inter-crossover or only infra-crossover.

2.3.3.5 Implementing GA without Crossover

In general, most GA research has been focused on using the crossover operator in developing advanced strategies instead of on mutation. However, since crossover is only a subset of all random mutations, the mutation operator is the most important operator that leads to population diversity. Jenkins (2002) proposed an adaptive GA process that used only mutation. In his research, two kinds of mutation were presented. One was random mutation and the other was intelligent mutation. Random mutation was similar to those used in traditional GAs, in which the mutation is performed with genes that are selected based on the mutation probability. In comparison, intelligent mutation was conditionally performed. For instance, if the stress in a truss member exceeded the design stress, positive mutation was applied to increase the cross section area. In the opposite case, negative mutation was applied. In addition, an upper boundary condition (ubc) and a lower boundary condition (lbc) on the objective value were defined as shown in Fig. 2.9. All individuals below the lbc were replaced by a corresponding number of higher ranked individuals similar to elitist strategy of traditional GAs. Individuals over the ubc were free from mutation as shown in Fig. 2.9. This adaptive

GA without the crossover produced benefits in that only a single population was able to be maintained through all generations, and memory requirements could be reduced.

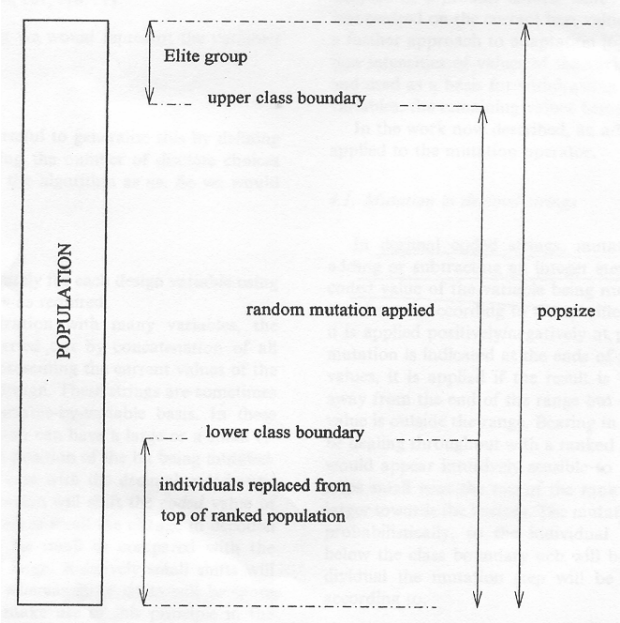


Fig. 2.9. Evolutionary processing of ranked population (Jenkins 2002)

2.3.3.6 GA Combined with Fully Stressed Design

In General, GA is a computationally expensive procedure. The search process is slower once the population starts to converge and it is often difficult to obtain the global optimum using GAs even though they are very robust in optimizing discrete domains. Yeh (1999) proposed a hybrid GA to enhance the efficiency of the GA process. Pure fully stressed design (FSD) is performed by a redesign rule, in which the member sizes of a truss are changed based on initial seeds until the stresses in the members are maximized. Therefore, FSD depends on an initial value and are very efficient in finding

local optimum. On the other hand, GA is better at finding the global optimum and is relatively independent of initial values. In the research by Yeh (1999), these two optimization methods were combined to enhance the search process. Although this research focused on the efficiency to speed up the search process using the hybrid-GA, the results from tests with 41, 50, and 72 bar trusses showed that the trusses obtained were more optimal than obtained using just FSD. In addition, the process time for convergence was superior to the time required by the GA and FSD methods alone.

2.3.3.7 Condensation Heuristic Methods

Long string lengths make the GA process slower in general. This is a serious problem, especially for large-scale optimization problems that require working with long strings. To overcome the problem of large-scale GA operation, it was proposed in research conducted by Jenkins (1997) to divide each individual into parts, and then operate crossover part by part as shown Fig. 2.10.

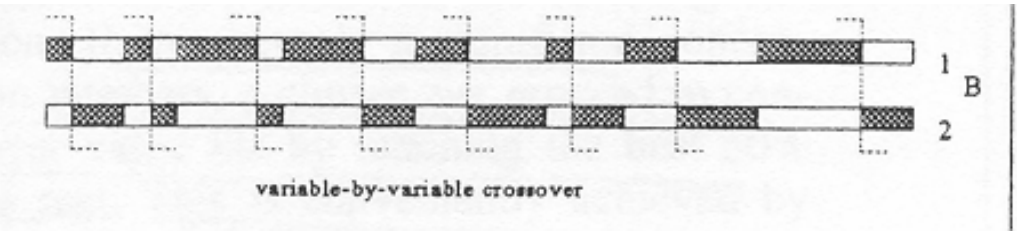


Fig. 2.10. Variable by variable crossover (Jenkins 1997)

In addition, by saving and updating the record of selection in every generation, regions were removed that were explored repeatedly and found to have low fitness. This strategy resulted in reducing the search space explored as shown in Fig. 2.11. After several generations, feasible, highly-fit individuals existed in the population. The condensation heuristic method was helpful in optimizing a structural design problem that had many objectives and penalties. The results obtained showed that the condensation method and advanced crossover operator efficiently produced more optimal results.

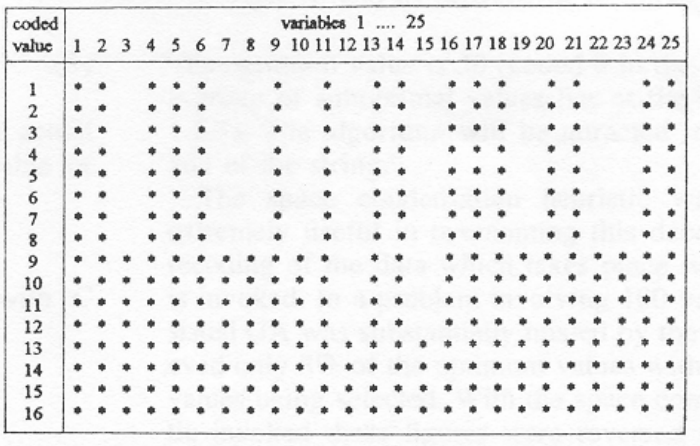


Fig. 2.11. Reduced search region, where (*) indicates available values (Jenkins 1997)

2.3.3.8 MOGA Using Fuzzy-Logic Penalty Function

In constrained multi-objective optimization, the fitness of each infeasible individual is penalized using penalty functions. This reduces the reproduction probability for the individual. By this strategy, individuals that are closer to the feasible

area and optimal Pareto set have higher fitness values. However, an individual that is closer to feasible area may be possibly dominated by an individual that is far from the feasible area due to the penalized fitness value, which causes improper searching in the constrained problem domain. In research conducted by Cheng and Li (1997), a strategy using fuzzy-logic penalty functions was presented. The fuzzy logic approach was first proposed by Zadeh (1965). Instead of using penalized exact values, the fitness used to calculate rank is penalized by an approximate value in proportion to the distance to feasible area as shown in Fig. 2.12. In the research, 4-bar and 72-bar trusses were optimized using the fuzzy-logic penalty function and a Pareto set filter. The results obtained showed that the fuzzy-logic penalty function resulted in a broader search of the problem domain by providing more qualify information concerning both the feasible and infeasible areas of the search space.

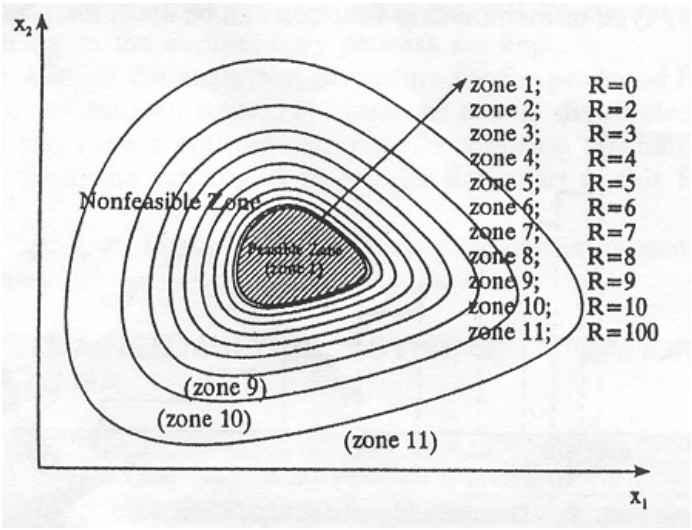


Fig. 2.12. Penalty function using fuzzy-logic (Cheng and Li 1997)

2.3.3.9 Multi-Objective Optimization Using Single Solutions

Coello and Christiansen (2000) proposed a new multi-objective optimization approach. In this research, the fitness function was defined as the combination of a weighted method and a min-max method for a single solution. The smallest value of the increments obtained from all the objective function values are calculated using a min-max method as follows (Coello and Christiansen 2000):

$$\text{Min-max method: } \min[\max\{z_i'(\bar{x}), z_i''(\bar{x})\}] \tag{2.4}$$

where

$$z_i'(\bar{x}) = \frac{\left|f_i(\bar{x}) - f_i^0\right|}{f_i^0}, \quad z_i''(\bar{x}) = \frac{\left|f_i(\bar{x}) - f_i^0\right|}{f_i(\bar{x})}$$

$$f_i(\bar{x}) = \left[f_1(\bar{x}), f_2(\bar{x}), \ldots\ldots, f_k(\bar{x})\right]^T$$

f_i^0 = attainable minimum of $f_i(\bar{x})$

Using the min-max optimum, Coello and Christiansen (2000) defined the weighting min-max method as follows:

$$\text{Pure weighting method : } \sum_{i=1}^{n} w_i f_i(\bar{x}) \tag{2.5}$$

$$\text{Fitness} = \sum_{i=1}^{n} w_i \min[\max\{w_i z_i'(\bar{x}), w_i z_i''(\bar{x})\}] \tag{2.6}$$

where

$w_i \geq 0$: weight coefficient

This multi-objective process was composed of several sub-processes. Each sub-process ran a separate single-objective GA. Using the fitness function given in equation 2.6 with weight vectors that were defined by user based on sparseness of solutions through wide

range of problem domain, each sub-process converged to a single solution. As a result of the entire process, the set of optimal solutions were formed by collecting the optimal solution from each sub-process. This approach was very simple because the obtained optimal set was the collection of the single solutions in each region indicated by the weight vector used.

2.3.3.10 Total Pareto Curve (Global Pareto Curve)

Using a structured genetic algorithm (Dasgupta 1994), Ruy et al. (2001) performed multi-objective optimization in a partially unstructured design. The problem domain was based on a ground structure. However, the goal was to find a total Pareto curve that is defined by several design alternatives simultaneously. The process consisted of two steps. In the first step, possible design alternatives were developed by switching each member's existence in the initial ground structure to achieve designs with varying topology. In the second step, the total Pareto curve was obtained by superimposing the individual Pareto curves obtained for each distinct truss topology through optimizing member size and geometry. Fig. 2.13 shows one of the results presented in the research showing the identification of the total Pareto-curve that was defined by different truss topologies. There is a transition that occurs in considering the objective tradeoffs where one topology is better than another. For meeting small displacement/higher weight objectives, Alt.C was optimal; while for meeting large displacement/lower weight objectives Alt.B was more optimal.

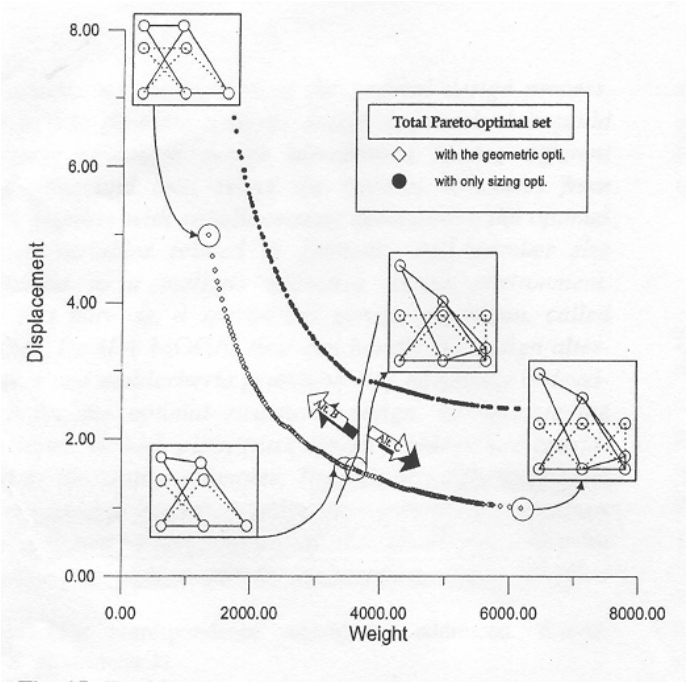


Fig. 2.13. Total Pareto curve with geometry design (Ruy et al. 2001)

2.4 Implicit Redundant Representation Genetic Algorithm

2.4.1 Introduction

GAs have been proven by various researchers to be very effective in searching discontinuous problem domains such as those defined in many structural design problems. Many of these studies, however, constrained the number of design parameters, which included the number of nodes and members. To expand the optimization search process to include diverse topologies and geometries simultaneously, a flexible problem domain representation that is able to synthesize design alternatives is required. This demand was satisfied by using the Implicit

Redundant Representation Genetic Algorithm (IRR GA), which was developed by Raich and Ghaboussi (1998). The IRR GA enhances the performance of a SGA by its ability to self-organize the GA representation and enable the individuals in population to encode a varying number of design parameters during optimization.

In a structured design problem domain, the number of parameters is explicitly bounded. To generate diverse topologies and geometries of trusses, however, an unstructured design problem domain, which has no explicit bounds in the number of parameters and can evolve diverse design alternatives, is required. The ability of the IRR GA to generate a varying number of design parameters in each individual in the population enables diverse truss design alternatives to be created and optimized in an unstructured design domain.

2.4.2 Description of the IRR GA

In a simple genetic algorithm (SGA), each design parameter is encoded into n-bit binary numbers. Each individual is constructed by concatenating the encoded design variables into a string. Consequently, these strings can be directly decoded into its corresponding phenotype, which is expressed by the design parameter values. In the IRR GA, however, redundant segments that do not encode any useful information for the phenotype are included as shown in Fig. 2.14. The redundant segments are similar to the presence of introns in biological genes. Each individual is composed of encoded sections and not-used sections. This new representation for GA individuals creates and

destroys design variables and provides a flexible representation for unstructured design problems (Raich 1999).

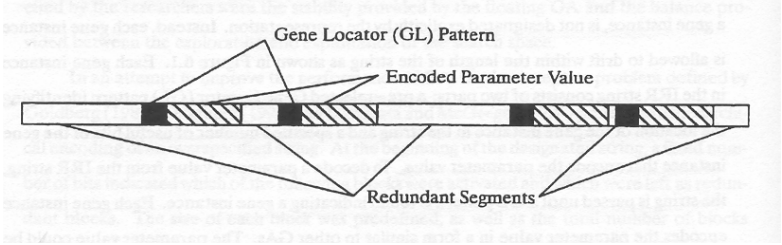


Fig. 2.14. Representation format (Raich 1999)

The encoded portion, which is called a gene instance, encodes the information for the phenotype. The locations and the number of gene instances are not pre-specified in the IRR GA individuals. Instead the location number can dynamically change by the action of crossover and mutation during search along with the redundant segments (Raich 1999). The IRR GA individuals include additional information encoded by redundant segments, which can possibly become gene instances in later generation. In this way, the number of parameters is not fixed in IRR GA. Over the generations, however, both the number and the values of encoded design parameters change in order to explore fitter areas of the search domain. This dynamic capability enables design synthesis to occur in an unstructured problem domain. Raich (1999) summarized the benefits of IRR GA over other GAs including SGA (Goldberg 1989) and structured GA (Dasgupta 1994):

1. Redundant segments in individuals leads to higher diversity in the population

even after the maximum fitness is reached.

2. Mutation will not only affect the binary valued bits contained in the gene instances, but will also affect the redundant segments.
3. Disruption of building blocks caused by crossover will be reduced due to the possibility of crossover occurring in the redundant segments.
4. The number of parameters represented is not explicitly encoded in the IRR GA and the number encoded by each individual in the population is allowed to vary.

2.4.3 Performance of IRR GA

The IRR GA performance was investigated by Raich (1999) on several defined problem domains, which were a mathematically-based deceptive problem, simple beam structural design problems, and a complex unbraced frame design problem. In the deceptive problem, the performance of IRR GA was compared to the performance of the SGA, structured GA and messy GA (mGA) (Goldberg, Korb, and Deb 1989). In the trials presented, the structured GA was not able to find the global optimum. Compared with the results obtained using SGA, several IRR GA trials found the global optimum in fewer than 20 evaluations while SGA required 40000 evaluations and mSGA required 40600 evaluations. In addition, the unbraced frame synthesis trials showed the benefit of IRR GA in supporting shape, geometry, and topology optimization in an unstructured domain without requiring any ground structure definitions for topology or geometry (Raich 1999).

SGAs provide robust and efficient methods for discrete problem structured domains. However, the lack of flexibility caused by the absence of redundant segments and fixed number of parameters in strings often makes SGAs inappropriate for unstructured problem domains.

3 DESIGNING A PARAMETER VALUE REPRESENTATION AND MODELING AN UNSTRUCTURED DOMAIN

3.1 Introduction

In order to design and analyze a truss, all the parameters that define the location of the nodes and member connections, and member sizes of a truss must be encoded in an individual. In addition, information concerning the support locations and loading conditions are essential in defining the unstructured problem domain. Therefore, to generate trusses in the unstructured design domain, the GA population must provide all the information required to define a diverse set of complete truss structures to evaluate each generation. To define each truss alternative, each individual's binary string is decoded into the design variable values. A predefined mapping between the binary representation and the design variable values, which called a design grammar, is defined. The design grammar used by the GA can strongly influence its performance since the search size and fitness landscape of the problem domain are determined by the selected design grammar. In addition, the proportion of feasible area to infeasible area is also determined by the design grammar. Therefore, the design grammar should be carefully defined in order to be suitable to explore a wide variety of potential design alternatives, while also being able to obtain near-optimal designs for these alternatives.

Many GA representations have been investigated with respect to being able to provide a flexible encoding of the design variables. As discussed in Section 2, a ground structure approach used a predefined topology and geometry for fixed numbers and

locations of nodes. Each member's presence/absence is determined by an 'on/off' bit (1/0 bit) in the GA individual. Limited geometry optimization has also been performed the locations of some nodes as design variables during shape optimization (Rajan 1995). In the research performed by Roston and Sturges (1996), a triangular element that represented a sub-geometry of a simple bridge was used. A number of triangles were joined at their baseline and the top nodes of each triangle were connected to generate a stable structure. In addition to developing a flexible representation and design grammar, the problem domain must be defined since the individuals (truss design alternatives) in the GA population must compete based on fitness in their environment, which is defined by the problem domain. If no limits are specified for the problem domain, the size of the search space is infinite. It would be useless to explore the search space for solutions using any computational optimization methods. In a practical design of a truss structure, the span length, maximum height of a structure, and other constraints can be prescribed to create boundaries that reduce the search domain size a little.

The objective of this research is to obtain a near-optimal set of trusses within a given problem domain by synthesizing design alternatives that have diverse shapes and topologies. To achieve the goal of this research, a flexible GA representation and design grammar are required, along with a defined unstructured problem domain. This section discusses the proposed design grammar for synthesizing and optimizing truss designs and the defined representation for the IRR GA that facilitated the flexible synthesis of design alternatives. The unstructured problem domain is modeled to practically limit the search space for the ease of adaptability of the results obtained to practice.

3.2 Design a Parameter Value Representation for IRR GA

The general format of the IRR GA (Raich 1999) was presented in Fig. 2.14. A single individual is composed of redundant segments and gene instances, which consist of the gene indicator pattern and the encoded parameter values. The gene instances contain the essential information, which is used to generate a truss alternative based on the decoding process defined by the design grammar. Unlike the structured GA (Dasgupta 1994), the locations of the gene instances are not explicitly specified in the IRR GA individual. The redundant segments in structured GA have their own fixed locations of encoded parameter values and redundant segments. By using control genes (on/off switches), the encoded parameter values are switched into redundant segments or vice versa. Therefore, the redundant genes in the structured GA are not able to affect other instances or redundant segments in individuals. Instead in the IRR GA, the locations can change and are therefore dynamically determined by the IRR GA itself. This feature provides flexibility and also allows the search process to be improved by possibly storing information in the redundant segments of an individual that may be used later. Fig. 3.1 shows an example of decoding the truss information from an IRR GA individual.

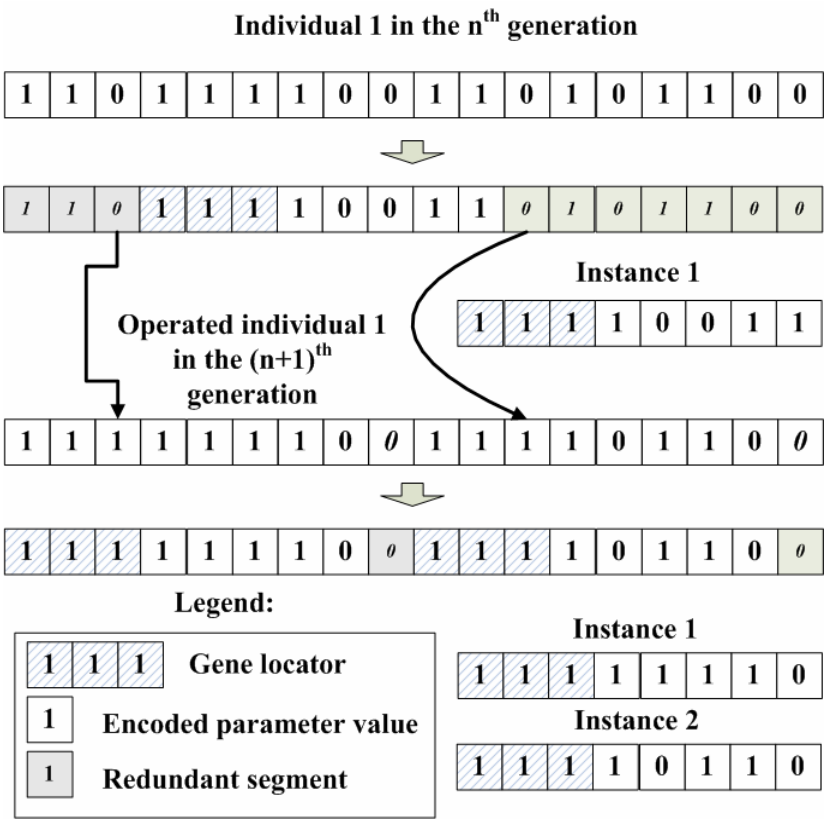


Fig. 3.1. Example of decoding in IRR GA

The location of an instance is designated by the gene locator pattern [1 1 1], which is indicated by the filled pattern in the Fig. 3.1. Instance 1 originally has one gene instance encoded in the n^{th} generation. The information contained in the single gene instance will be decoded using the defined design grammar. In the $(n+1)^{th}$ generation, the values of two bits are changed from 0 to 1 due to the actions of crossover and mutation. The result is that Individual 1 now encodes two instances that contain information. Even though the redundant segments defined in Individual 1 are not used in the n^{th} generation, Individual 1 in the $(n+1)^{th}$ generation is strongly affected by these

redundant segments becoming active and encoding design variable information.

Another benefit is that the IRR GA optimizes the number of variables as well as their values. In structural design problems, each design alternative considered may have a different topology and/or geometry, which means that each design alternative has a different number of design variables that must be encoded to contain information concerning nodal locations, member sizes, etc. By using the IRR GA representation developed in this research, various design alternatives were able to be synthesized effectively to meet the research goal of optimizing structural designs in an unstructured domain.

As mentioned previously, IRR GA optimizes the number of variables as well as their values. This means that the number of instances in each individual in a GA population can not be fixed. To generate trusses using the information encoded in a varying number of instances, each instance should have information that is independent of other instances. To complete a stable structure, each instance should have relative values in order to establish the relationships that exist with the other independent instances. In this research, trusses were generated based on information of nodes concerning nodal locations. Fig. 3.2 shows the design variables defined based on the truss nodes. The nodal information includes the x-y coordinates, the number of connections to that node, and the cross section areas of each of the connecting members.

In the IRR GA representation, one gene instance indicates encodes one node and the other required nodal information. Each node is connected to other nodes based

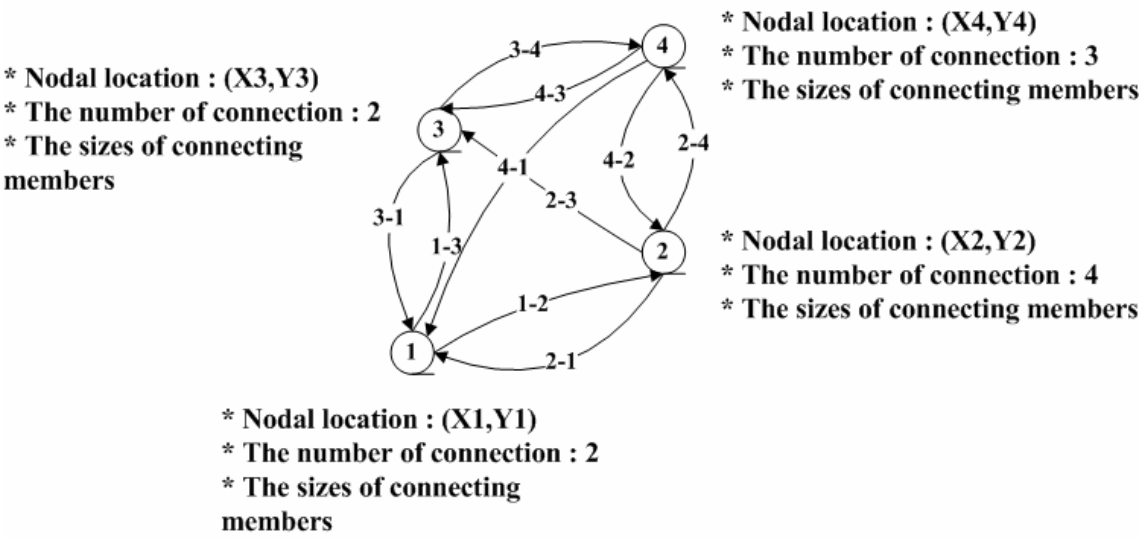


Fig. 3.2. Connections of nodes based on nodal information

on the number of connections indicated in the encoded nodal information. In cases when the number of generated nodes is less than the number of connections (Node 2 in Fig. 3.2), the number of connections is set equal to the number of existing nodes. The connection of nodes proceeds in the order of shortest distance with the sizes of cross section indicated in nodal information. By encoding node-based information and considering each node as an independent object, the proposed IRR GA representation and design grammar are flexible enough to represent diverse truss alternatives. The performance obtained by the IRR GA in synthesizing and optimizing alternatives benefits from the flexibility provided by changing the number of instances in IRR GA individuals. Fig. 3.3 shows the format of the IRR GA individuals defined in this research effort. Each gene instance contains the nodal-based design variable information

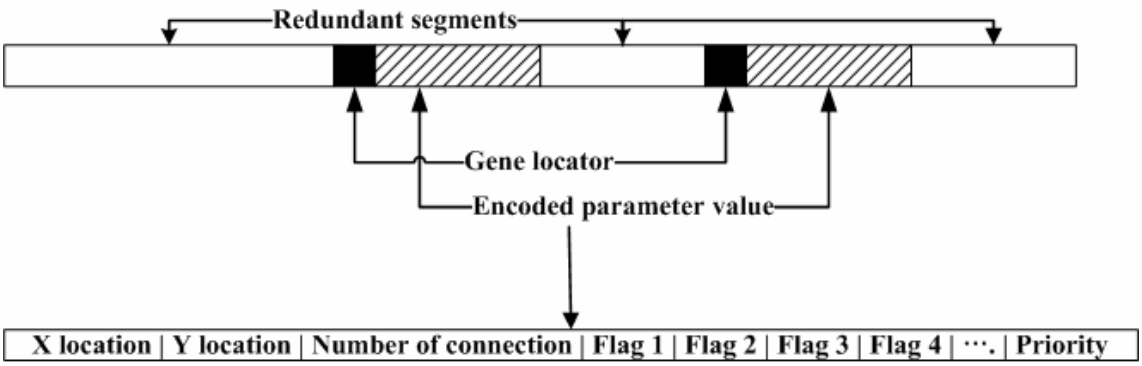


Fig. 3.3. Format of IRR GA string

, which includes the nodal location, the number of connections, the cross-section areas of connecting members, and a priority. Instead of specifying the cross-sectional areas as continuous values, a set of steel W-sections were selected from AISC LRFD Manual (AISC 2001). Flag values were used to indicate the selected members. Each encoded flag value is an integer that designates the section area of each created member. As shown in Fig. 3.3, more than a minimum number of flag values are encoded. When the individual is decoded, a flag value is matched to each member connection specified. For instance, the first generated truss member, which connects the two nodes that are closest in distance, is assigned to the property of the pre-selected member designated by first flag (the left flag), and then the second generated member is assigned the property designated by second flag. Encoding flag values that can handle the number of connections designated helps in evolving design alternatives during exploration of the search domain.

Since GAs provide a stochastic method, several situations may occur during the

decoding of the string that need to be avoided. Therefore, several exception clauses are defined in the design grammar. The representation for the IRR GA proposed in this research defines that one gene instance should define one node. In certain individuals, there may occur an encoding where several instances indicate the same nodal location, even though the other information such as the number of connection and member section sizes differs among the gene instances. In other words, one node has overlapped information for itself even though each node should have one set of information for generating a truss. Two methods were considered to solve the overlapped node problem. First, the phenotype of a gene instance containing the nodal information is modified to avoid the overlap. In this case, the nodal locations are modified arbitrarily, or by some specific values, to make better results intentionally, which is called a repair strategy. The modified phenotype will become different from the original phenotype that was decoded from the IRR GA gene instance. This mismatch between encoded and decoded information may possibly mislead the IRR GA during the search process since the objective values are computed using the phenotype. Therefore to prevent this mismatch from occurring, the gene instance in the IRR GA should be modified to make them match with the modified phenotype.

Another method that could be used is to select only one gene instance for each node and ignore the rest of overlapped nodal information. The selection of which information to keep is determined by the IRR GA. The concept of diploid strings is presented in order to implement the second method. A single-stranded individual has all the information required to define a complete phenotype. It is the simplest type of

genotype found in nature. To build more complex plants and animals in nature and to increase their adaptability against the environment, however, a more complex type of genotype is needed. The diploid form of the genotype has one or more pairs of chromosomes. Unlike haploid (single-stranded individual), the diploid genotype includes more information than needed to define the essential information to form a phenotype. In Fig. 3.4, the capital letters indicate dominant genes. As a result, the pair of genes AA expresses as A in phenotype. In a similar manner, bB is B, CC is C, Dd is d, and ee is e. Assuming that D is blonde hair and d is black hair, one of these feature will dominate since a human can not have both blonde and black hair. The offspring of this person could possibly have either blonde or black hair because their parents had both genes. As described in the example, the conflict of redundant information is solved by the dominancy genetic operator. In this case, only the information from the dominant gene is used in the phenotype. The mechanism that uses diploid individuals and dominance provides more flexibility and adaptability to environment by saving additional information for use perhaps in the future.

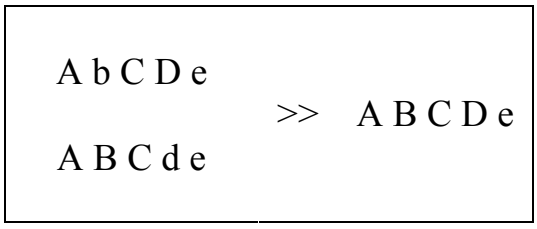


Fig. 3.4. Example of a pair of chromosome (Goldberg 1989)

To solve the overlapped node problem in this research effort, the concept of dominancy and diploid is used. As shown in Fig. 3.3, each individual gene instance encodes a priority value. In case that more than one gene instance indicates the same nodal location, only the nodal information with the highest priority among the overlapped nodal information is used to form the truss. If more than one gene instance has the same nodal location and the same priority, the first decoded instance is assigned the highest priority.

The priority value of the node is also used to solve the overlapped connection problem that may occur. As described previously, a node is connected to other nodes in the order of closest distance by the number of times the connection number designates in a gene instance. Due to the definition, more than one connection is able to exist between two nodes. Fig. 3.5 shows an example of the overlapped connection problem, which is removed using the priority value. Node 1 is connected to Node 3, and at the same time, Node 3 is connected to Node 1. This decoding results in an overlapped

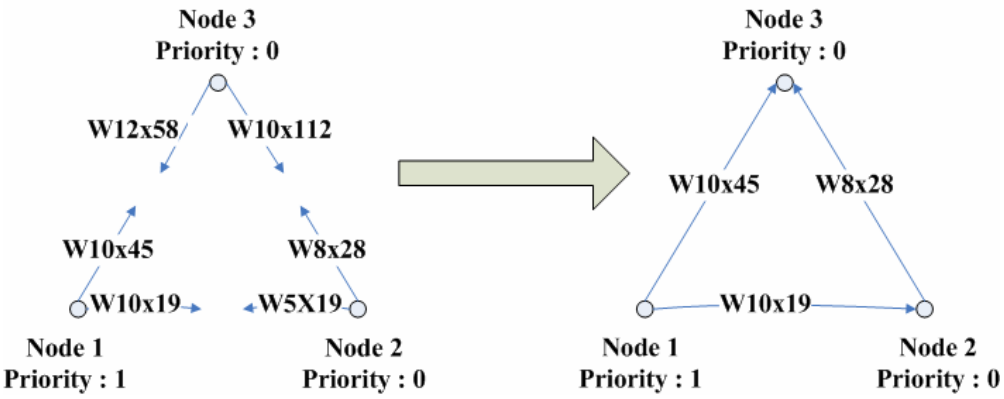


Fig. 3.5. Connecting nodes and imposing member-sizes using the priority values

connection between Node 1 and Node 3. The connection from the node with the higher priority is only expressed to form a truss. The other connection is removed from the truss. However, the IRR GA individual still has connection information encoded from instances with lower priorities. This mechanism provides the instances with a chance to be expressed the phenotype in later generations. Implementing the diploid and dominancy enabled the IRR GA to encode more information than the essential information for a phenotype. This improved the flexibility and adaptability of the IRR GA to evolve solutions in the problem domain.

The problem domain defined in this research allows the definition of all possible trusses, including unsymmetrical trusses. This research focuses on symmetric truss structures. Constraints that require symmetry of members and nodal locations could be defined. Instead, in this research, the IRR GA generates only half of structure. Then by overlapping the generated information to the other side, a complete truss structure is generated. The reduced problem domain resulted from using the imposed symmetry saves computational expense. Fig. 3.6 presents an example of the sequence through which the encoded truss information is used to define a stable truss. The flexibility provided by the proposed IRR GA representation will create a large search space in the design domain, which means that a number of unstable trusses and other types of undesirable structural designs may be generated. This is the tradeoff that must be accepted, however, to allow a diverse range of truss designs to be defined in the problem domain. In order to meet the objective of this research generating a number of possible design alternatives is necessary. Therefore the proposed design grammar and

GA representation are considered as suitable to perform the optimization in this research effort.

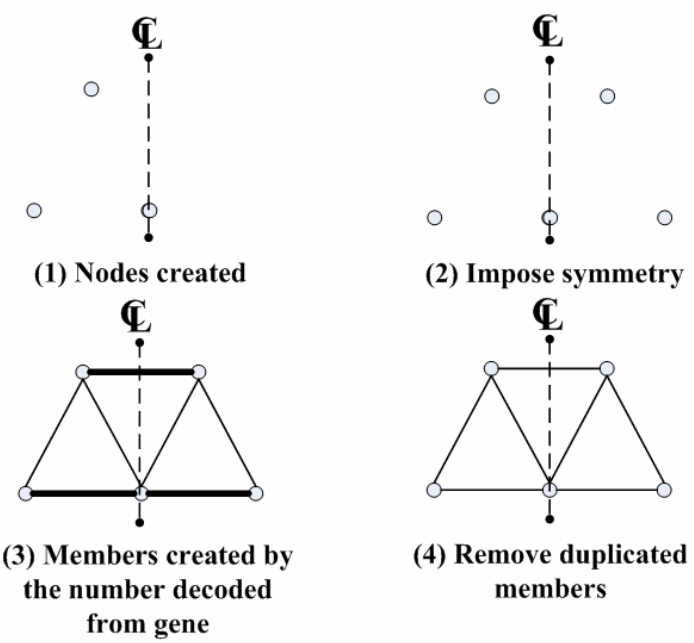


Fig. 3.6. Truss generation using the nodal & member information decoded from the IRR GA genotype

3.3 Modeling the Unstructured Domain for the Truss Design Problem

As described in Section 3.1, unstructured design problems have no explicit bounds placed on the number of design parameters. The unstructured design domain reflects a realistic design domain that has no pre-knowledge concerning information of design parameters. A structured design domain, in comparison, assumes values for many parameters and is required to work with fixed information of design parameters. For evolving truss designs, the unstructured domain has no predefined information

concerning the number of nodes and members, member sizes, and nodal locations. In reality, the combination of the continuous value and discrete value for design parameters creates the large and complex design domain. Therefore, minimal design information is specified for the unstructured domain in order to limit somewhat the design domain In this research, the minimal design information is limited to:

1. *Maximum height of the truss.*
2. *Span Length.*
3. *Support conditions.*
4. *Loading.*

Fig. 3.7 presents a view of the unstructured design domain used to create truss structures in this research.

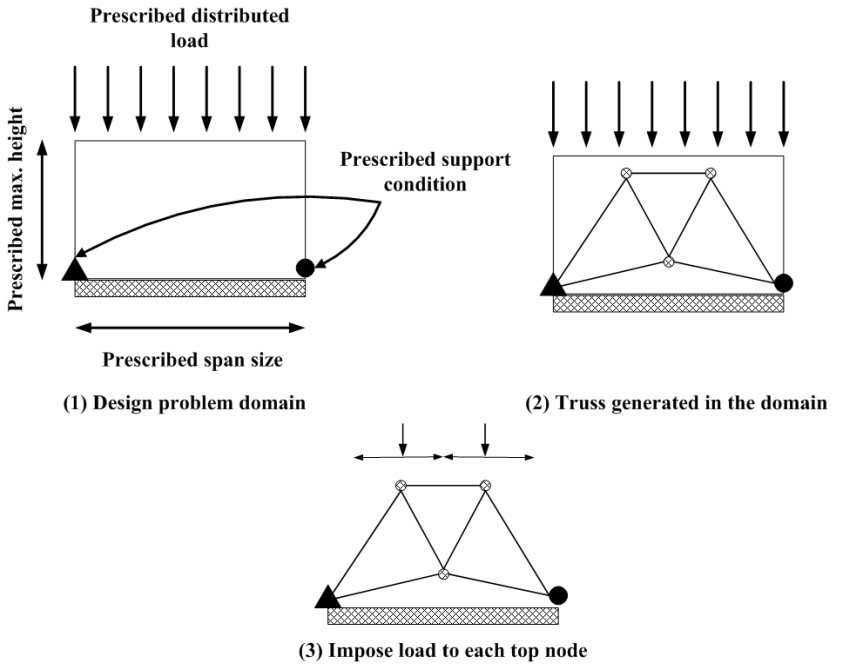


Fig. 3.7. Unstructured design domain and imposed load conditions

In the unstructured problem domain, the number of nodes and their locations are allowed to vary. Therefore the load is assigned as distributed load instead of point load on specified nodes. When the design parameters are decoded from the IRR GA individual to obtain the nodal locations then the distributed load is converted into point loads on each top node of the truss based on tributary area. The prescribed design information used to investigate the proposed optimization method in this research is:

1 Maximum height of trusses: 15 ft.

2. Span Length: 40ft., 60ft., and 80 ft.

3. Support Conditions: Hinge and Roller (simple truss)

4. Load: 5 kips/ft.

All the research performed concerning the performance of the proposed representation and strategies will use these values to define the unstructured design problem domain.

4 MULTI-OBJECTIVE OPTIMIZATION USING A COMPOSITE FITNESS FUNCTION

4.1 Introduction

In the structured problem domain, single-objective design optimization has been performed by other researchers and shown efficient results. For example in sizing optimization that seeks to minimize the total weight of a truss, the optimization process changes the size of truss members within a pre-selected group of member sizes. Because the designs are optimized based on a fixed topology and geometry in a structured design domain, the single-objective optimization process will result in stable and efficient trusses. In an unstructured design domain, however, the geometry and topology of the structure are not predefined. If a truss is optimized in this domain using only a single objective of minimizing weight, the optimization process will probably generate low weight trusses that have no members or they are not stable. In addition for unstructured domains, specifying multiple objectives that conflict with one another prevents the IRR GA from obtaining a solution that is extremely partial toward optimizing only one objective. In this case, multi-objective optimization is essential to explore design alternatives in an unstructured problem domain.

In the real world, many engineering problems are subject to multi-objective optimization. In addition, these objectives are usually conflicting and non-commensurable. For example in optimizing a construction management problem, conflicting objectives occur since as the duration of construction decreases, the direct

cost of construction increases and the indirect cost decreases. To obtain the optimal duration of a construction project, multi-objective optimization should be performed considering the two objectives of time and cost. The relationship between the two objectives is shown by the optimal curve of solutions that show the tradeoffs that occur. These results can assist engineers in better managing efficiently a construction project. Consider another example in structural optimization. Decreasing the weight of a structure causes the deflection of the structure to increase. The tradeoff relationships that exist between the weight and deflection objective of structures can also help design engineers select the design alternatives that are most appropriate for their design project.

There are several approaches to performing multi-objective optimization. One is to select a single objective to optimize and regard all the other objectives as constraints. In GAs, this method is performed by penalizing the single objective value using penalty values that reflect how well the other objectives are satisfied. Another approach is to combine all the objective values into one function called the composite fitness function. By maximizing or minimizing the composite fitness function, each objective can be optimized. The drawback of these two approaches is that they both model the original multi-objective problem stated in an inadequate way by aggregating the objectives linearly or non-linearly (Coello and Veldhuizen 2001). For an example in design problems, both of the approaches discussed generate a single near-optimal solution. If the goal is to explore design alternatives, however, obtaining a single solution is nowhere near what the design engineers desire. Therefore, to meet the designers' needs many trials would need to be performed empirically in which each trial modifies the

balance between the objectives.

A more beneficial approach for design problems using GAs is to obtain an optimal set of solutions by determining dominated and non-dominated individuals. By computing and comparing the multiple objective values for each individual in population, dominated and non-dominated individuals are determined. The non-dominated individuals form the Pareto-optimal set of solutions. For design problems, the optimal set of designs provides design engineers with a selection of diverse design alternatives. In addition, comparing the engineer's designs with the Pareto-optimal set, the quality of their designs can be evaluated.

In this Section, the objectives and constraints are defined for the conceptual design and optimization of roof truss structures. Using the identical objectives and constraints, an effective composite fitness function is empirically formulated that is relatively easy to computationally implement. The roof truss problem domain defined by the IRR GA representation proposed in Section 3 is investigated using the composite fitness function in the IRR GA. To obtain a consistent high-level of performance from the IRR GA, the effect of the probability rates of crossover and mutation are investigated. In addition, the required IRR GA string lengths for three truss span lengths (40 ft., 60ft., and 80 ft.) are determined based on maintaining truss complexity and population diversity. The IRR GA parameters and composite fitness function defined in this Section will used to perform multi-objective optimization (MOGA) by ranking the population and using Pareto-optimal concepts to evolve a set of near-optimal designs in Section 5.

4.2 Define Objectives and Constraints for Roof Truss Design

In a truss design problem, the factors that should be considered in trying to achieve an optimal truss design are the total weight, the maximum deflection, and the magnitude of stress in the truss members. The final goal of this research on truss optimization is to produce economically efficient designs. Optimization is performed considering the weight of the structure, since for trusses especially it is a larger portion of the construction cost. To meet the serviceability requirements, the deflection of the truss structure should be considered and limited based on designer specified deflection limits. If the structure was well optimized considering only minimizing weight, the large deflection that would probably occur makes the truss useless in practice. In addition, the maximum stress in each of the truss members must be considered. High stresses in each member under assigned loads typically indicate the efficiency of the structure in carrying the loading. Fig. 4.1 presents typical tradeoff relationships between these objectives. Increasing the truss weight causes both the truss deflection and the member stresses to be decreased. In other words, weight and deflection are conflicting objectives in the optimization process, as well as weight and stresses. Decreasing the weight of a truss can also cause the stresses in the truss members to exceed the maximum stress allowed considering buckling issues for compression members. Trusses that have overstressed members should be excluded from consideration as final design alternatives.

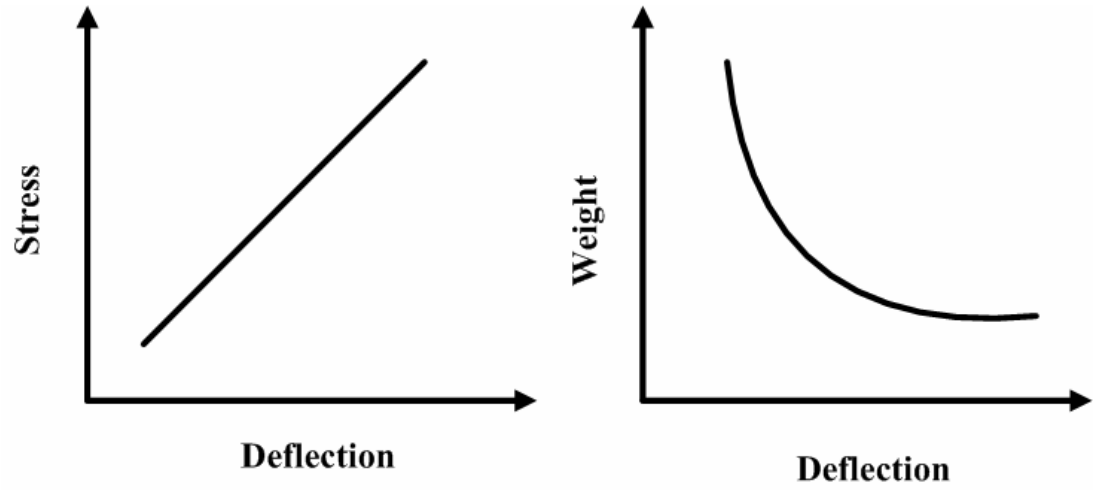


Fig. 4.1. Stress vs. deflection and weight vs. deflection

As defined in the IRR GA representation presented in Section 3, a nodal location can be generated anywhere within the defined problem domain. Having a very short member connected between two nodes that are close will cause difficulty in construction. Having too long a member because nodes are very far apart will produce moments in the truss members due to self weight and also result in vibration problems due to member slenderness. Considering construction requirements and the serviceability of the truss structure, the distances between each connected node, which is the length of the members, should be limited.

Table 4.1 lists the objectives and constraints defined for the research presented. The weight of the truss was based on a density of 490 lb/ft^3 (7849 Kg/m^3) and the material used was A36 steel with a yield stress is 36 ksi (248.22 N/mm^2). to calculate the stresses and deflections in the truss. The modulus of elasticity, E, was 290,000 ksi (2x10^6 N/mm^2). In this Section, the deflection of a truss was defined as the average

value of each nodal deflection in a truss. The minimum length of a truss member was limited to 5ft. and the maximum length to 20 ft. For each of the trials performed, ten W-sections were selected from LRFD (AISC 2001) that were W10x112, W18x71, W8x67, W12x58, W10x45, W8x28, W8x24, W10x19, W5x19, and W6x16.

Table 4.1. Objectives and constraints

Objectives	1. Total weight of a truss 2. Deflection of a truss * Average deflection of the truss nodes (except support nodes)
Constraints	1. Stability * $m+3-2j \geq 0$ * $0 <$ Deflection $<$ defined max. deflection limit 2. Stress * actual stress $\leq$ allowable stress (considering buckling) 3. Member length * maximum length of a member is 20 feet * minimum length of a member is 5 feet

4.3 Format of IRR GA Gene Instance and Formulation of Composite Fitness Function

In the Fig. 3.3, the general format of an encoded gene instance is presented. The maximum number of connections that was allowed from each defined node was initially determined as four. Therefore, four independent members' section sizes were encoded to the gene instances as flags. Table 4.2 presents the number of binary bits required to encode each design variable. As shown, the maximum height of any truss

Table 4.2. Definition of number of bits used to encode all design variables

Design variable	Required bits
x-coordinate	40 ft. span: 0<x<20: 5 bits 60 ft. span: 0<x<30: 5 bits 80 ft. span: 0<x<40: 6 bits
y-coordinate	Maximum height of a truss: 15 ft.: 4 bits
Number of connection	Number of connection for each node: 2~4: 2 bits
Flag 1	10 member sections selected from LRFD: 4 bits
Flag 2	Same as flag 1
Flag 3	Same as flag 1
Flag 4	Same as flag 1
Required total bits for one instance	40 ft. span: 27 bits 60 ft. span: 27 bits 80 ft. span: 28 bits

was limited to 15 ft. The number of connections from each node was required to have a value between two and four because every node in a truss should have at least two connections to provide stability. The nodal coordinates were encoded using integer values, therefore the precision of the coordinates was one foot. Consequently, 27 bits are required for each gene instance in order to encode all the nodal and accompanying member information.

4.4 Formulation of the Composite Function

Genetic algorithms are fundamentally influenced by the probability of performing mutation and crossover during the optimization process. In addition, since a multi-objective optimization problem is formulated using a composite fitness function, the formulation of the IRR fitness function also influences the performance of the IRR GA. Both of these effects on performance are highly empirically determined.

A composite fitness function can be formulated in general as composed of fitness and penalty terms.

$$\begin{aligned} F &= \sum f(x) \quad \text{or} \quad F = \prod f(x) \\ P &= \sum p(x) \quad \text{or} \quad P = \prod p(x) \end{aligned} \tag{4.1}$$

As shown in equation 4-1, the fitness and the penalty function generally are formed by adding or multiplying several fitness values or penalty values together. Each form of the fitness function has its own benefits. For example, the contribution from each objective value or penalty value are magnified more by multiplying each term together than by

adding each term. The effect also depends on the relative values between the objectives and penalties. Therefore, the formulation of fitness and penalty functions must be determined for each particular problem. In equation 4.1, f(x) indicates an objective function and F indicates a composite fitness function. The increment of the value of f(x) leads to the increments of the composite fitness. To minimize each objective, the composite fitness must also be minimized. GAs, however, work by maximizing the fitness function and by seeking the fitter individuals. Thus, the composite fitness function should be converted into a maximization function:

$$F_M = C_F - F \quad \text{or} \quad F_M = \frac{C_F}{F} \tag{4.2}$$

where

C_F = Constant to ensure the generation of only positive fitness values.

In GAs using a composite fitness function, constraints are applied typically by reducing the fitness of each individual by subtracting or dividing the fitness by the penalty values. This makes any individual that violate constraints look worse in the solution process. Consequently, the total composite fitness function can be defined as:

$$F_{TOT} = \frac{F_M}{P} \quad \text{or} \quad F_{TOT} = F_M - P \tag{4.3}$$

From equations 4.2 and 4.3, the decrease in the objective and penalty values causes an increase in the total composite fitness.

In this research, several types of composite fitness functions were investigated

to attempt to obtain a function that resulted in better performance consistently for the truss problem domain. Table 4.3 outlines the definitions of the fitness and penalty functions used in this research. Each objective and penalty value is normalized by using the previous maximum objective and penalty values. Because the range of value in each objective and penalty is different from one another, the normalization of objective and penalty values will prevent IRR GA individuals from being selected based on fitness that is partial to one objective or penalty value.

All the investigations to determine the total composite fitness function were performed in the domain of 40 ft. span, and the total string length was arbitrarily determined. In addition, the initial values of other GA parameters were randomly determined and modified through the trials. As a first trial, the total composite fitness function was formulated by adding the fitness and penalty functions.

$$F_{TOT} = C_T - (F_W + F_D + P_S + P_{DS} + P_{st}) \tag{4.4}$$

where

C_T = Constant to ensure the generation of only positive fitness values
For the trials, $C_T = 50$

Table 4.3. Fitness and penalty functions corresponding to each objective and constraint

Objective 1	$F_w = \frac{W_i}{Max.W}$	W_i: total weight of i[th] truss in the population $Max.W$: maximum value of W in the previous iteration
Objective 2	$F_D = \frac{D_i}{Max.D}$	D_i: average deflection of i[th] truss in the population $Max.D$: maximum value of D in the previous iteration
Constraint 1	$P_s = \frac{S_i}{Max.S}$	**Stability constraint** $S_i = \|m+3-2j\|$ or 1 : errors in analysis $m+3-2j$ is equal to or larger than zero than $S_i=0$ $Max.S$: maximum value of S in the previous iteration
Constraint 2	$P_{DS} = \frac{DS_i}{Max.DS}$	**Member length constraint** $DS_i = \frac{\sum_j^{N_i}(LM_j - 20) + \sum_j^{N_i}(5 - SM_j)}{N_i}$ N_i: number of a truss members in the i[th] truss LM_j: length of j[th] member in the i[th] truss $(LM_j-20)<0$ then $LM_j-20=0$ SM_j: length of j[th] member in the i[th] truss $(5-SM_j)<0$ then $5-SM_j=0$ $Max.DS$: maximum value of DS in the previous iteration
Constraint 3	$P_{st} = \frac{ST_i}{Max.ST}$	**Stress constraint** $ST_i = \frac{\sum_j^{N_i}(\sigma_j - \sigma_{cr})}{N_i}$ N_i: number of members in the i[th] truss σ_j : stress of j[th] member in the i[th] truss σ_{cr} : allowable stress $(\sigma_j - \sigma_{cr}) < 0$ then $(\sigma_j - \sigma_{cr}) = 0$ $MaxST$: maximum value of ST_i in the previous iteration

Fig. 4.2 shows the number of individuals that violated each constraint and that satisfied all the constraints. In this trial, the probability of crossover is 0.5 and the probability of mutation is 0.05. Over 90 percent of individuals violated both constraint 1 and 2. This result indicates that most of the individuals in the population are unstable and the truss members indicated are too long or too short. The IRR GA optimization process using equation 4.4 was not able to find even one individual that satisfied all the constraints, let alone try to search for the fittest individual and an optimal solution.

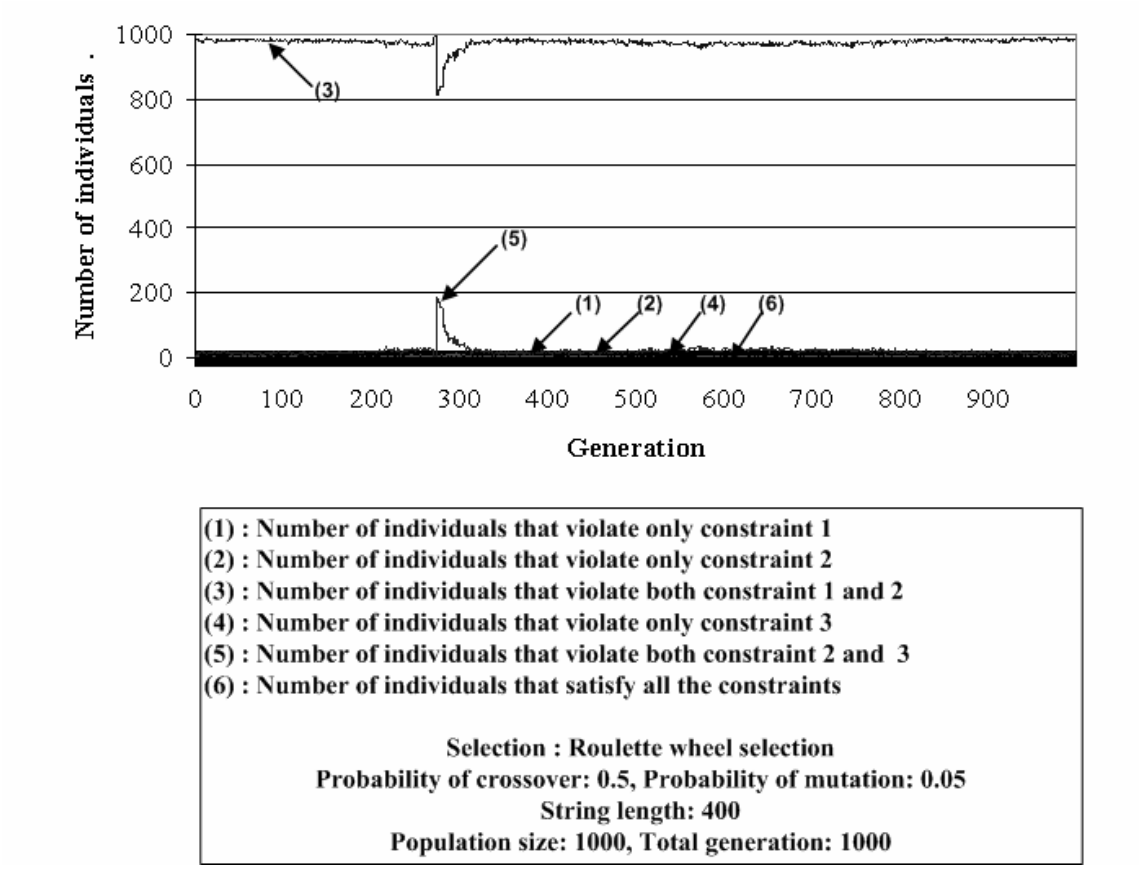


Fig. 4.2. Distribution of individuals in the population using equation 4.4 for 40 ft. span trusses

To try to force individuals in the IRR GA population to the feasible areas of the search space, a different total composite fitness function was investigated. In equation 4.5, the values resulting from adding the fitness values together is divided by the sum of the penalty values.

$$F_{TOT} = \frac{C_T - (F_W + F_D)}{(P_S + P_{DS} + P_{st} + 1)} \tag{4.5}$$

where

C_T = Constant to ensure the generation of only positive fitness Values,
For the trials, $C_T = 50$.

Thus, the variation of total composite fitness is less affected by the objective values and is chiefly affected by the values of penalties. The expectation is that this will force the IRR GA to place a higher priority in minimizing the penalty values, which means fewer constraints violations. Fig. 4.3 presents the results obtained for the IRR GA using equation 4.5. Compared with the previous result, the individuals in the population are relatively well distributed in the infeasible areas. In addition, a small number of individuals satisfy all of the constraints, which means no penalties.

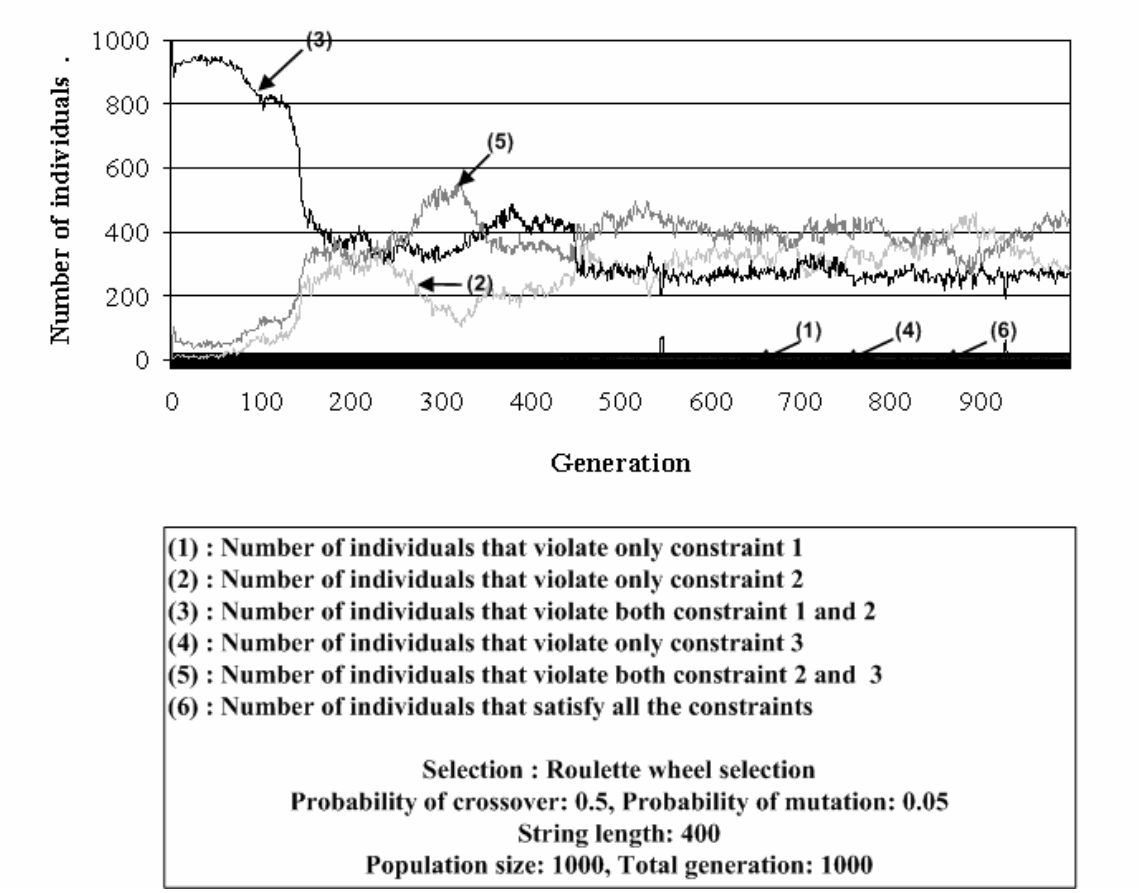


Fig. 4.3. Distribution of individuals in the population using equation 4.5 for 40 ft. span trusses

To enhance the result, the constants values in equation 4.5 were modified.

$$F_{TOT} = \frac{C_T - (F_W + F_D)}{(C_S P_S + C_{DS} P_{DS} + C_{st} P_{st} + 1)} \tag{4.6}$$

where

C_T, C_S, C_{DS}, and C_{St} = Constants,
C_T=50; C_S= 10; C_{DS}= 100; C_{st}= 50.

The constants in equation 4.6 were empirically determined based on the distribution of

individuals over the entire search domain. Fig. 4.4 shows the results obtained using equation 4.6 in the IRR GA. Approximately 20 percent of the individuals exist in the feasible area. Among all of the investigations performed, the total composite function presented by equation 4.6 showed the best performance. The fitness function 4.6 presented tended to strongly force the individuals into the feasible area. Even though the strong force caused the individuals to converge early, this control was necessary due to

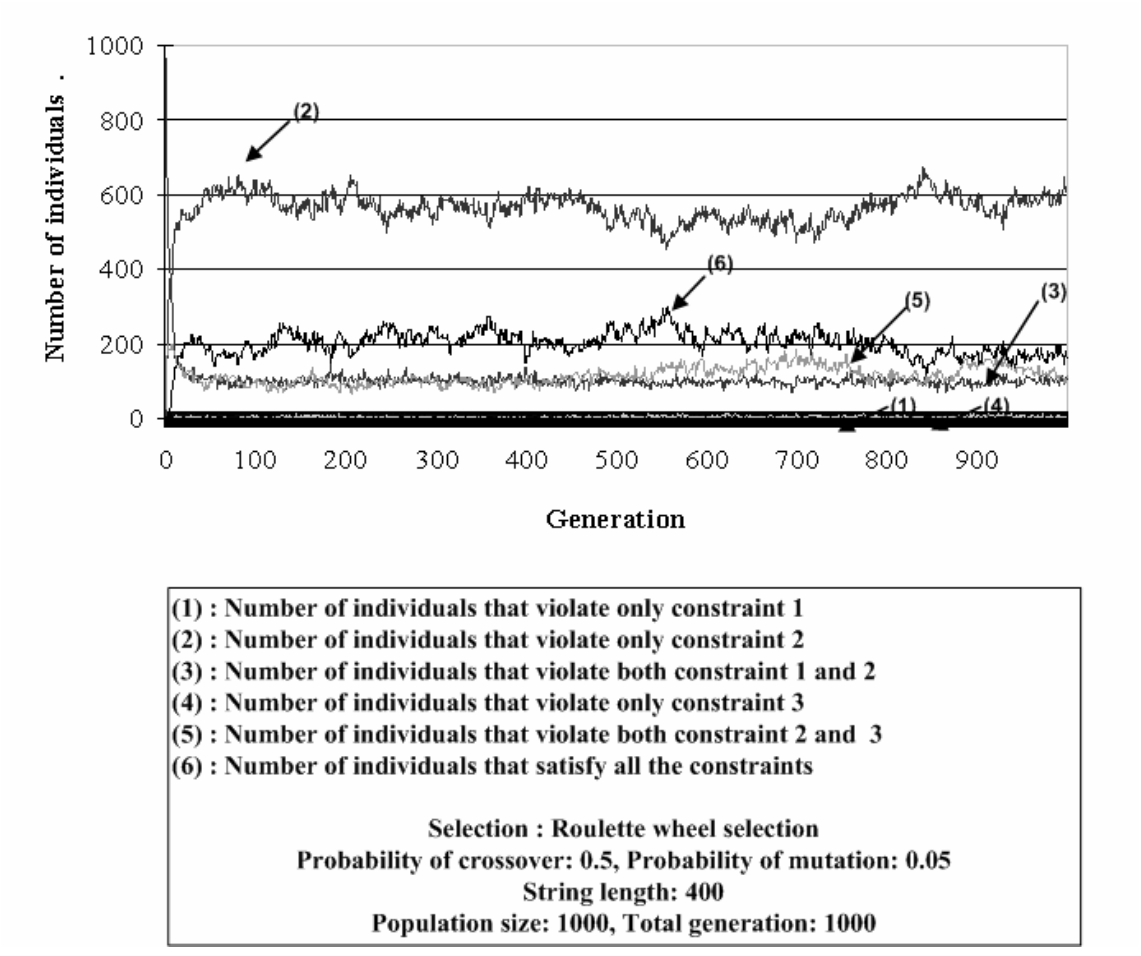


Fig. 4.4. Distribution of individuals in the population using equation 4.6 for 40 ft. span trusses

the very large infeasible area defined by the problem domain. The fitness function (4-6) was used in all further investigations in this Section with some modifications of parameters.

4.5 Determination of the Required IRR GA String Length for each Truss Span Length

In the IRR GA, the total string length strongly influences the role redundancy plays in the optimization process (Raich 1999). As mentioned previously in Section 3, the diversity of design alternatives in this research is the direct result of redundancy of IRR GA representation. A longer GA string will include more redundant segments, which facilitates the role of redundancy. In other words, a longer string enables the IRR GA to generate more diverse design alternatives. However, the longer string length expands the search space that must be explored by the IRR GA. Using the specified IRR GA representation proposed in this research, the longer string can generate greater number of nodes, which causes the average distances between the nodes to be shorter. Consequently, the shorter distance between nodes will increase the possibility that more individuals violate the constraint that limits the minimum length of truss members. To provide a balance between the diversity of the design and the size of search space, the optimal IRR GA string length must be investigated.

In this research, 40 ft., 60 ft., and 80 ft. span trusses are investigated. For each design domain, optimization trials with different string lengths were performed. In all these trials, the probability of crossover and mutation were set at the same values. In

addition, two-point crossover was used and tournament selection was adopted with a tournament size of 4 individuals. The initial redundant ratio can be calculated as following (Raich 1999):

$$N_0 = \frac{(l_s - l_g + 1)p}{(1 + l_g p)} \tag{4.7}$$

where

l_s= the number of bits in the string minus (n-1) bits to account for the end of the string,
l_g= the number of bits in an encoded gene instance,
n= the number of bits specified in the gene location (GL) pattern,
p= the probability of a single occurrence of a specific GL pattern,
N_0= the total number of instances of GL pattern and encoded genes.

The probability of a single occurrence of a specific GL pattern is calculated as following (Raich 1999):

$$P = \beta^n \left[\frac{1-\beta}{1-\beta^n}\right]^2 \sum_{j=0}^{n-1} \beta^j \tag{4.8}$$

where

β = the probability of a single occurrence of the specified bit value,
N = the number of bits specified in the GA pattern.

For the GL pattern of [1 1 1] specified in this research, the probability of the occurrence can be calculated as p=0.07142857. Table 4.4 shows the initial numbers of instances for each string length in 40 ft., 60 ft., and 80ft. span which are calculated using equations 4.7 and 4.8.

Table 4.4. The initial number of gene instances for each string length

String Length	Number of bits in encoded gene instance	
	27	28
200	4	4
300	7	6
350	8	8
400	9	9
450	10	10
500	12	11
550	13	12
600	14	14
650	15	15
700	16	16

Fig. 4.5 presents the number of feasible individuals obtained for each string length investigated in the 40 ft. span truss domain. As presented in Fig. 4.5, curves (1) and (2) show the highest number of feasible individuals in the IRR GA population. This means that the shorter the string length, the greater the chance that more of the population in the feasible region.

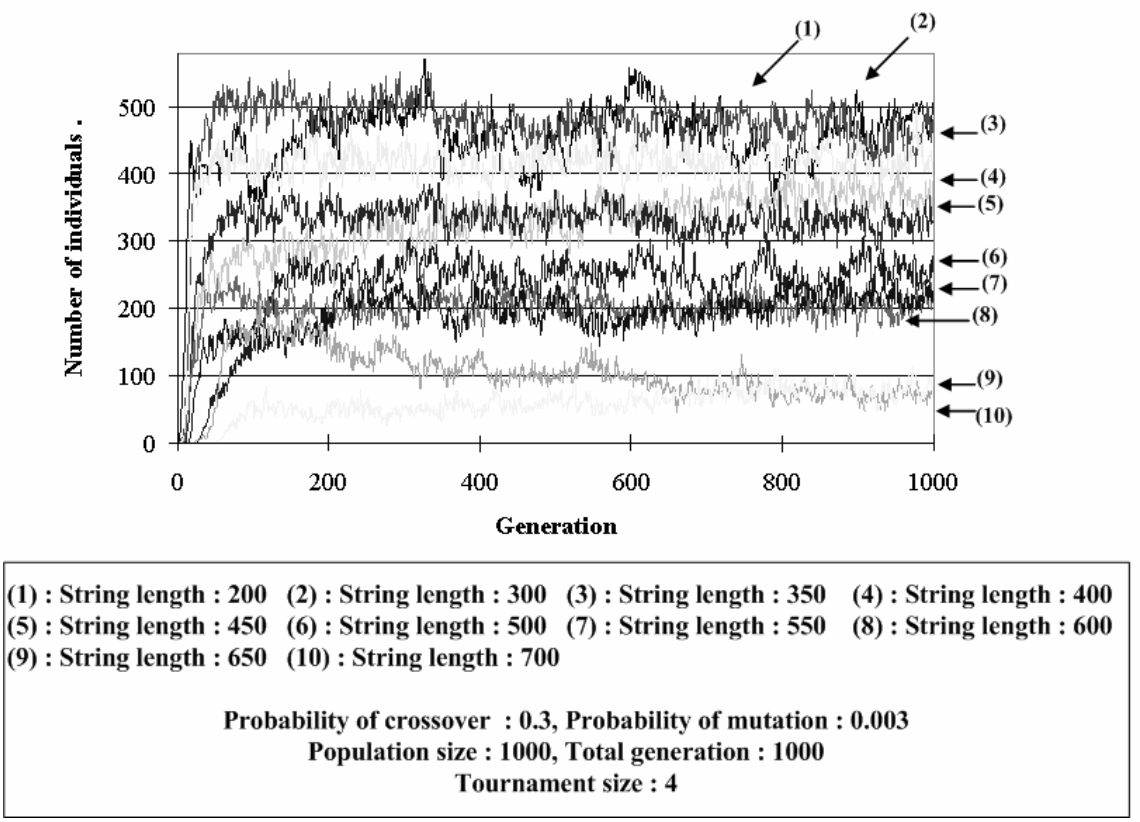
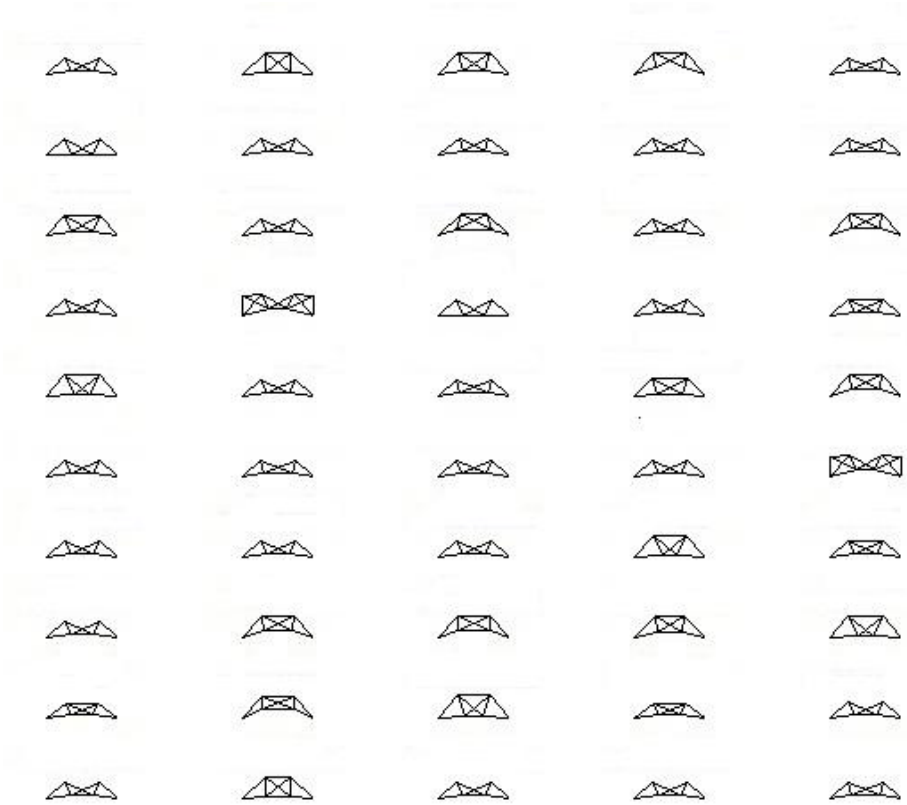
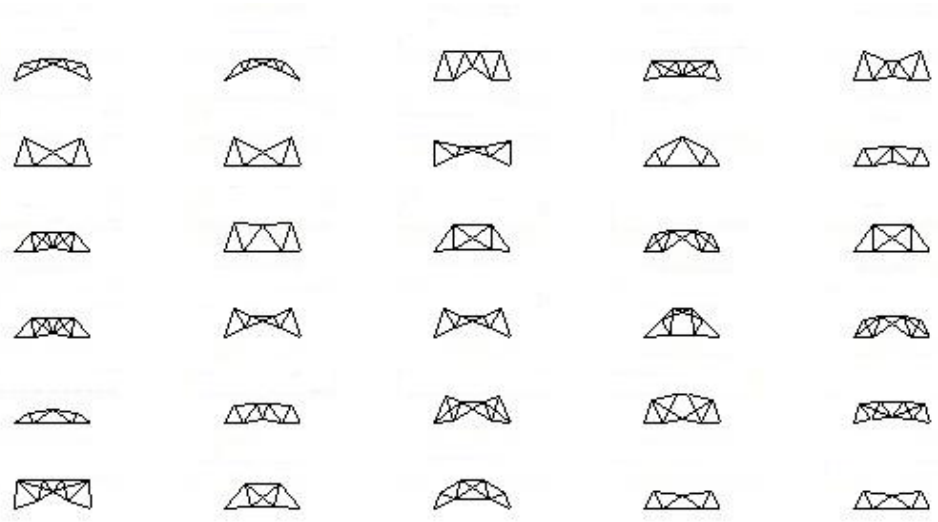


Fig. 4.5. The number of individuals in the feasible area for different string lengths for 40 ft. span trusses

In addition to the number of feasible individuals, the extent to which the diversity of design alternatives in population is maintained also should be verified to achieve the objective of this research effort. Fig. 4.6 shows a subset of trusses from the population that satisfies all the constraints in an early generation for string length of 200 and 300 bits. In the 10^{th} generation, individuals with a 200 string lengths show a lack of diversity in the truss topologies created. In comparison, individuals with a 300 string length show more diverse shapes and topologies.



(a) Feasible individuals at 10^{th} generation with 200 string length

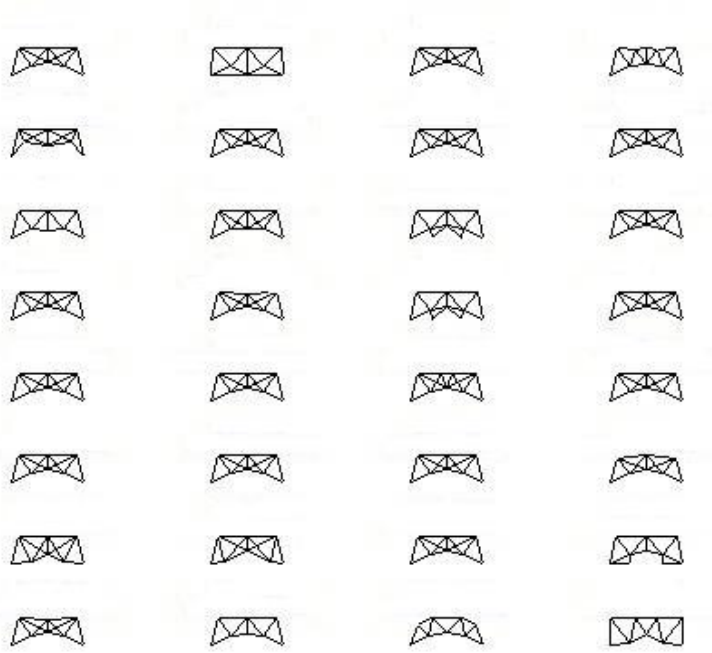


(b) Feasible individuals at 10^{th} generation with 300 string length

Fig. 4.6. Trusses in populations optimized using two different IRR GA string lengths

In the case of individuals with longer string lengths, all the feasible individuals were rather similar in truss geometry and topology. Fig. 4.7 presents a view of the trusses generated using a 700 bit string length in two different generations of the same IRR GA trial. In the 16^{th} generation, only 5 trusses satisfied all of the constraints in the population. Thus the importance of diversity cannot be discussed with this small number of trusses. In the 30^{th} generation, however, the trusses evolved still show a lack of diversity in truss shapes and topologies even though more redundancy exists in the longer string. One reason that can be inferred from this result is that long strings create a large search space, which causes the IRR GA to require more iterations to generate many feasible individuals. In addition, the proposed composite fitness function imposes a bias that strongly pushes individuals into the feasible area. This strong pressure toward feasible area causes the individuals in the population to converge early and maintaining any diversity would be difficult. Consequently, before the IRR GA begins to generate a large number of diverse feasible individuals, many of the population individuals are expected to converge to a certain local area and to cause the monotony in truss topologies.

(a) Trusses that satisfy all the constraints in 16th generation.



(b) A portion of trusses that satisfy all the constraints in 30th generation.

Fig. 4.7. Trusses generated with 700 string length in different generation

Fig. 4.8 presents the fittest trusses in the population obtained after 1000 iterations for trials using different string lengths. The fittest individuals obtained resulting from the composite fitness function have different truss topologies

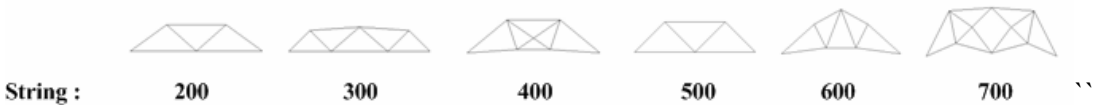


Fig. 4.8. The fittest trusses produced with different string lengths

corresponding to each string lengths. The overall trend is that as the string length increased, the truss complexity also increased. In addition to being sensitive to the string length, the results from the IRR GA were found to be extremely sensitive to other parameters, including the probabilities of crossover and mutation, the number of crossover points, and the constants specified in the fitness function. Thus, any changes in the IRR GA parameter values during investigations usually caused a change in the results obtained. However, a larger number of feasible individuals and more diverse truss topologies were maintained in the population using the 300 string length than with other string lengths.

To investigate the performance of the IRR GA and composite fitness function further, trials were performed using the same procedures and same parameters used for the trials in the domain for 40 ft. span trusses for 60ft. and 80 ft. span trusses. Fig. 4.9 presents the feasible individuals generated using different string lengths in the domain for 60 ft. span trusses. Among the 10 trials performed with different string lengths, the three that resulted in superior performance are shown. Curve (1) in the Fig. 4.9 has mostly higher values than the other two curves. The values shown on curve (1) are approximately between 100 and 200 throughout all the iterations. Compared with the results obtained for the 40 ft., span, the number of feasible individuals overall decreased. As the span length increases, however, the number of design alternatives certainly increases. In other words, increasing the span length implies the expanse of the search domain. The expanded search domain causes the IRR GA to need more generations and a larger population size than the search domain defined for a shorter span.

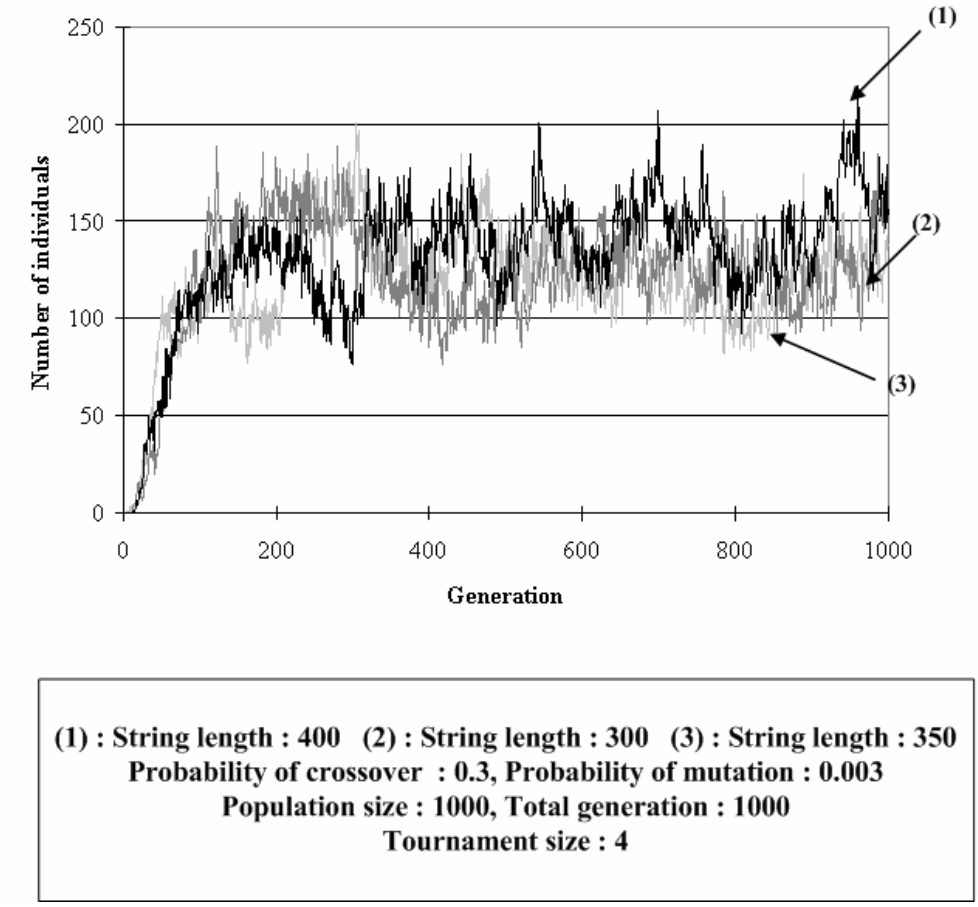


Fig. 4.9. The number of individuals in feasible area with different string lengths for 60 ft. span trusses

Therefore, with the same IRR GA parameters, the number of feasible individuals decreases in the domain with a longer span size. The result presented in Fig. 4.10 show a decrease in the number of feasible individuals found in the domain for 80 ft. span trusses. The approximate range of the curve (1), which is the result from the process with 600 string length, is between 15 and 40 feasible individuals, which is from 1.5 to 4 percent of the total population.

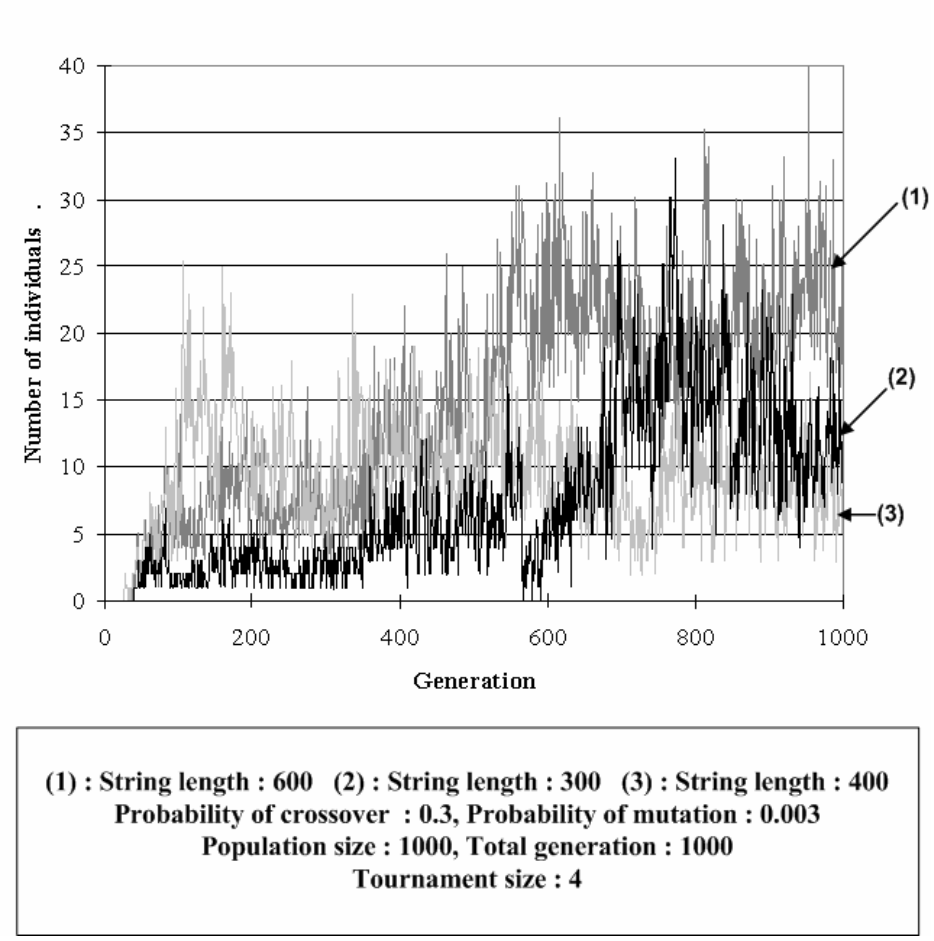


Fig. 4.10. The number of individuals in feasible area with different string lengths in the domain for 80 ft. span trusses

The diversity of truss geometry and topology in a population was investigated along the string lengths that resulted in a higher number of feasible individuals in IRR GA population. In the trials for 60 ft. span trusses, most of the individuals maintained less diverse topologies as compared with 40 ft. span trusses. There were no remarkable differences in the diversity of trusses that were produced by the different string lengths. In the trials for 80 ft. span trusses, the diversity could not be directly evaluated due to

the small number of feasible individuals found in each generation. In addition, most of individuals in population showed complex topologies of trusses, and the number of feasible individuals for the 80 ft. span is lower than for the 40 ft. and 60 ft. span.

Fig. 4.11 presents the fittest trusses evolved after 1000 iterations for different string lengths. The results for 40 ft., 60 ft., and 80 ft. span lengths showed that different topologies will be evolved corresponding to the string length used. The results from the IRR GA in the truss design domain defined were very sensitive to the GA parameter settings. Unlike the trials for 40 ft. span trusses, there was no single outstanding string length that resulted in more feasible individuals for trials with 60 ft. and 80 ft. span length. To determine the most efficient string lengths for these spans, additional investigations were performed. In each generation, the two objective values, which were the total weight and the average value of nodal deflections, were compared to one another.

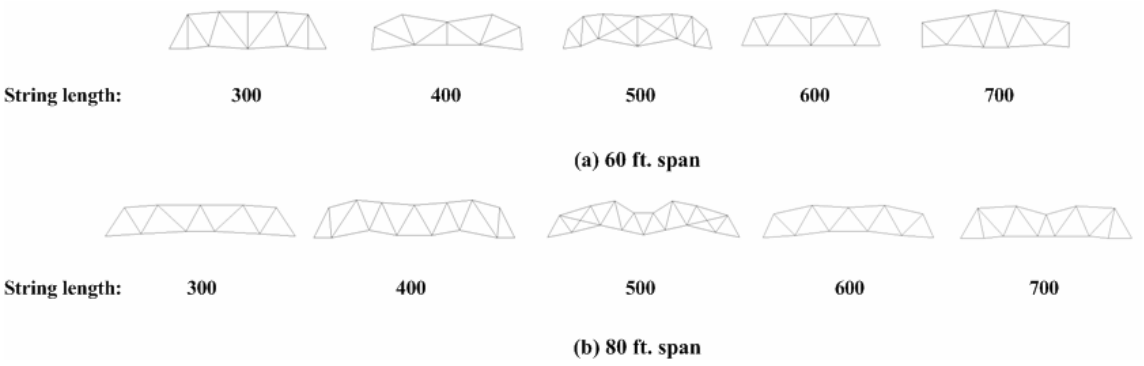


Fig. 4.11. The fittest trusses for each string length

The dominant trusses, in which both weight and deflection are lower or at least one objective is lower than in others, were saved through all iterations. Fig. 4.12 presents the dominant trusses obtained from the IRR GA trials performed with different string lengths. The curves, in which each point indicates a truss generated by the IRR GA, show the tradeoff relationship between deflection and weight. Among all of the results produced for the 10 different string lengths, three of the results that were the most critical are shown. In most regions of the Pareto-optimal curve, the results from the trial performed with 300 string length are the most dominant.

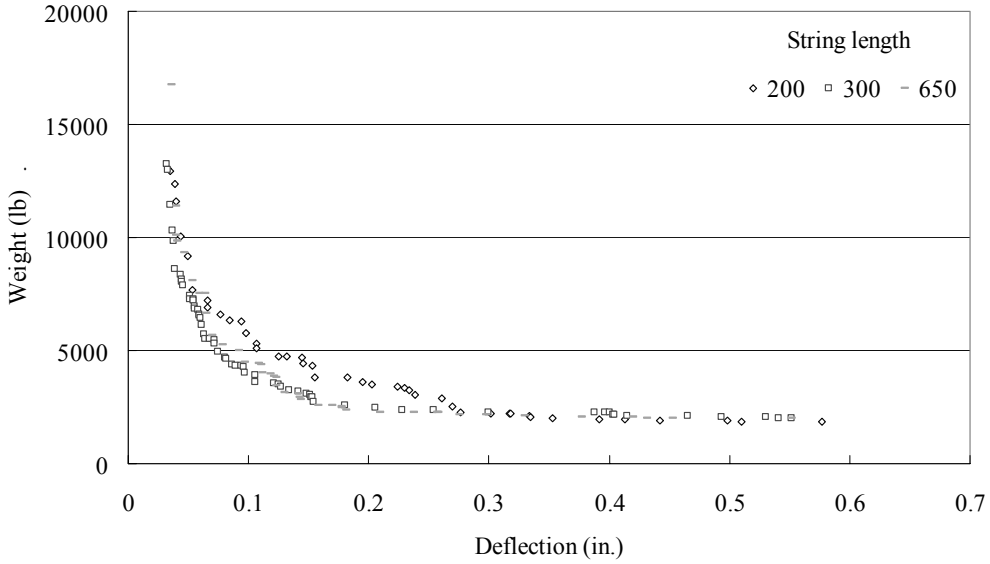


Fig. 4.12. Dominant individuals for each string length for 40 ft. span trusses

Among the results obtained from nine IRR GA trials performed using different string lengths for the 60 ft. span length, the three most critical results are shown in Fig.

4.13. In most regions of the Pareto-optimal curve, the results using the 400 string length were the most dominant. Even though the previous investigations using 300, 350, and 400 string lengths showed that a similar number of feasible individuals were obtained and no outstanding differences in the diversity of truss topologies were found, the objective values produced using the 400 string length were dominant over the results obtained using other string lengths as shown in Fig. 4.13.

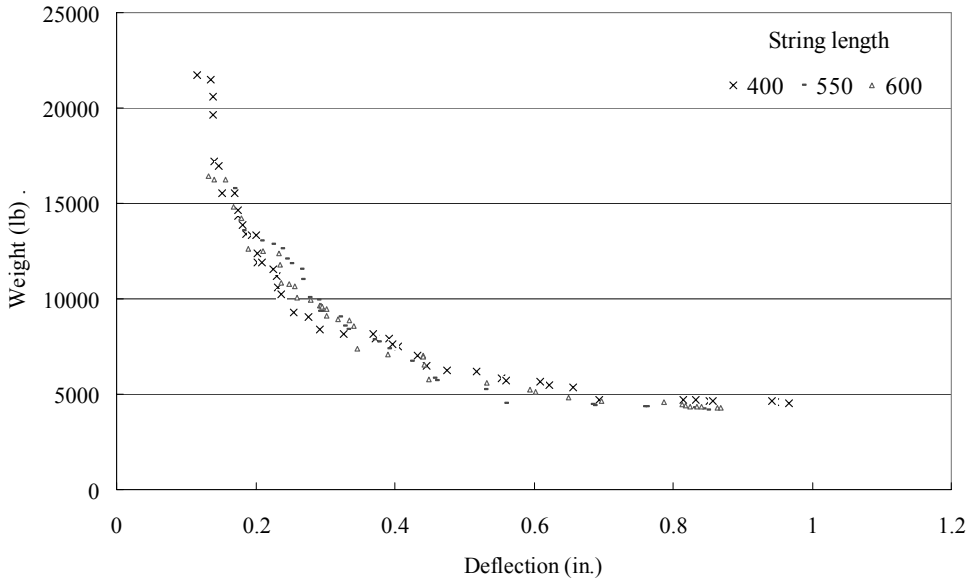


Fig. 4.13. Dominant individuals for each string length for 60 ft. span trusses

In a similar manner, the results obtained for 80 ft. span length trusses were observed. Fig. 4.14 presents the dominant individuals in each population for three different string lengths. Although the individuals obtained in trials for both 300 and 600 bit string lengths were found to be most dominant, the individuals obtained using the

600 bit string length comprised a wider range of the design domain than obtained using the 300 bit string length.

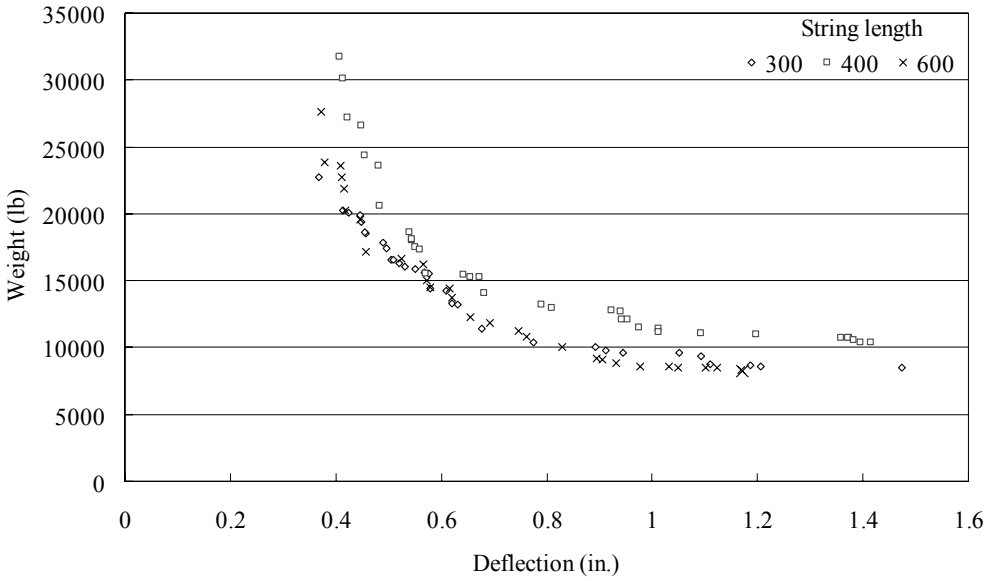


Fig. 4.14. Dominant individuals for each string length for 80 ft. span trusses

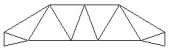
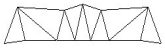
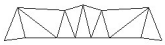
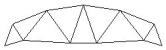
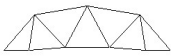
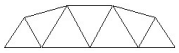
4.6 Discussion of the Effect of IRR GA Parameters and the Fitness Function

In this Section, the results obtained for multi-objective optimization performed using a composite fitness function were presented. The goal of the optimization performed was to determine the most efficient string length for each span length investigated, and in addition, to observe the characteristics of the problem domain created by the IRR GA representation proposed in Section 3.

The results obtained depended greatly on the IRR GA parameters. As seen in results obtained showing the fittest trusses for each span length, the fittest trusses had

different topologies depending on the string length used. In addition, the results were very sensitive to the changes in the probabilities of crossover and mutation during several trials. Table 4.5 presents a summary of the results obtained from trials performed using different parameters. The trials were performed for 60 ft. span trusses using a 400 bit string length. In Cases 1, 2, 3, and 4, the topologies of the fittest trusses were different corresponding to the probabilities of crossover and mutation. In addition, the distribution of feasible individuals and the diversity of truss topologies in population were different. In Case 2 and 3 where different probabilities of crossover and the same probability of mutation are applied, the fittest truss topology was similar; whereas the maximum number of feasible individuals in the two populations was different. Changing the mutation probability caused different truss topology to be found as the fittest and also increased the number of the feasible individuals in the population. As shown by Case 3 and 4, a small increase in the probability of mutation and a fixed probability of crossover causes a large difference in the number of feasible individuals in the population. The diversity of truss topologies in the population, however, significantly decreased. In Case 5, the number of crossover points was modified. Compared with Case 1, the number of feasible individuals decreased a little probably due to more disruption caused by more crossover points destroying good information in the individuals. In addition, the diversity of truss topologies also decreased. Cases 6 to 9 show the results of the trials performed with different constants in the composite fitness function.

Table 4.5. The results from different parameters

	Defined parameter values	The fittest truss shape	Max. # of feasible individuals
CASE 1	Pc: 0.8, Pm: 0.008, Ncp: 2 C_T: 100, Cs: 10, C_{DS}: 100, C_{ST}: 50		258
CASE 2	Pc: 0.5, Pm: 0.005, Ncp: 2 C_T: 100, Cs: 10, C_{DS}: 100, C_{ST}: 50		182
CASE 3	Pc: 0.3, Pm: 0.005, Ncp: 2 C_T: 100, Cs: 10, C_{DS}: 100, C_{ST}: 50		197
CASE 4	Pc: 0.3, Pm: 0.003, Ncp: 2 C_T: 100, Cs: 10, C_{DS}: 100, C_{ST}: 50		601
CASE 5	Pc: 0.8, Pm: 0.008, Ncp: 5 C_T: 100, Cs: 10, C_{DS}: 100, C_{ST}: 50		211
CASE 6	Pc: 0.8, Pm: 0.008, Ncp: 2 C_T: 5, Cs: 10, C_{DS}: 100, C_{ST}: 50		10
CASE 7	Pc: 0.8, Pm: 0.008, Ncp: 2 C_T: 100, Cs: 1, C_{DS}: 10, C_{ST}: 5		5
CASE 8	Pc: 0.8, Pm: 0.008, Ncp: 2 C_T: 100, Cs: 10, C_{DS}: 50, C_{ST}: 100		129
CASE 9	Pc: 0.8, Pm: 0.008, Ncp: 2 C_T: 100, Cs: 10, C_{DS}: 10, C_{ST}: 10		11

Notes:

Pc= Probability of Crossover
Pm= Probability of Mutation
CPoint= Number of Crossover points

Increasing C_T or decreasing the constant values for the penalty functions causes the sensitivity of total fitness function corresponding to the variation of penalty values to decrease, which results in placing the higher priority in minimizing the objective values rather than minimizing the penalty values. Consequently, there will be more individuals in the population that violate the constraints. As seen in Case 6, 7, 8, and 9, changing the constants affects the fittest truss topology and the number of feasible individuals. In Cases 6, 7, and 9, the decrease in constant values in the objective and penalty terms caused a low pressure on individuals being in the feasible area. Consequently, the number of feasible individuals was greatly decreased. In addition, the changes imposed in Case 8 caused different topologies to occur for the fittest truss. The changes of constants of total composite fitness function strongly affected the number of feasible individuals as well as the topologies of the fittest trusses.

Through investigating many trials with different parameters, consistent results were not obtained. A few changes in parameters leads the individuals in the population to a different area of the search space. As observed in many trials, the problem domain is considered intensely sensitive. In other words, the problem domain is extremely complicated. For instance, even a few changes in nodal locations will create a new configuration of a truss. The truss will be optimized using 10 member sections that are pre-defined. Consequently, small changes in the truss configuration will create another local optimal set of solutions. Allowing the simultaneous change in the geometry, topology, and member sizes of trusses generates an inconceivable large size of search domain that is composed of innumerable local optima and in the multi-objective sense,

multiple, equally-optimal solutions. In addition, the balance of distribution of the individuals over the search domain is affected by the constants specified in total composite fitness. For instance, the increase in constant used for constraint 1 will cause the decrease in the number of individuals that violate constraints in the population. In the same manner, the other constants will mainly affect the distribution of individuals in the search space. In this extremely complicated domain, it is impossible to figure out the optimal combination of constant values. Even if the optimal parameter values are determined, the changes of search domain resulting from the different span length or string lengths will probably make the parameters non-optimal. In addition, without any schemes that enable the individuals in population to be widely distributed all over the search domain, the optimization using the composite fitness function causes individuals to simply to fall into a single local optimum. Consequently, the effect of initial parameter values is maximized in these cases.

As stated previously, another problem of the multi-objective optimization by composite fitness function is that the single solution obtained from one trial may not be in the range of deflection and weight that design engineers desire to use. This drawback of the composite fitness function requires a large number of trials with each trial trying out a new balance of the objective values in order to meet the demands of design engineers.

To solve the above problems, multi-objective optimization should be performed by ranking and Pareto optimal methods. Moreover, strategies for the distribution of individuals across the Pareto-optimal front should be adopted. In Section 5, several

concepts and strategies proposed in previous research will be introduced and modified strategies will be proposed to assist in optimizing truss designs in the complicated search domain.

5 MULTI-OBJECTIVE OPTIMIZATION USING NON-DOMINATED PARETO OPTIMAL METHODS

5.1 Introduction

In Section 4, optimization using a composite fitness function was discussed. As stated previously, using a composite fitness function that is an aggregation of the multiple objective values results in convergence of the GA population to a single solution. As the number of objectives increases, the optimization for a single solution becomes difficult due to tradeoffs that exist among the objectives. In addition, the single solution obtained only reflects one possible weighting of the objectives. In design, the relative weighting of the objectives is impossible to do a priori. Therefore, obtaining a single solution will not necessarily reflect the designer's priorities. To address this problem, each objective should be optimized independently, instead of combining all the objectives into one measure. This approach enables the discovery of an optimal set composed of solutions that are 'equally' good at optimizing conflicting objectives specified.

The Vector Evaluated Genetic Algorithm (VEGA) was proposed by Schaffer (1985) as the first attempt to perform multi-objective optimization using GA to obtain a set of solutions. In VEGA, each individual in the new population was selected using proportionate selection based on how well each independent objective was met. The individuals selected based on each objective were shuffled before performing crossover and mutation operations. The results obtained by VEGA showed that the population

converged to regions related to each objective rather than the converging to all over the non-dominated Pareto-optimal surface. This result may cause the middle point solution, which is a very useful solution that is not biased to only one objective, to not survive because it does not have extreme values for any objective (Coello and Veldhuizen 2001).

Niche count, which is a measure how much an individual is shared in a population, enables GA individuals to be well-distributed all over the search by providing the information of how much an individual is shared in a population. In this research, niche count was implemented by using sharing functions (Goldberg 1989) and the strength (Zitzler and Thiele 1999).

5.1.1 Non-Dominated Pareto Ranking and Sharing Functions

Goldberg (1989) suggested non-dominated Pareto ranking, in which all the non-dominant individuals in the population are assigned the same fitness for selection. By comparing all the individuals with one another, the highest rank is assigned to the non-dominated individuals. Excluding these highest-ranked individuals, the same procedure is executed to assign the next highest rank to the non-dominated individuals remaining in the population. The procedure is repeated until all individuals are ranked. Non-dominated Pareto ranking provides a rational way to consider all the conflicting objectives that cannot be compared directly with each other.

To help distribute individuals over the entire Pareto-optimal surface, a sharing function can be applied. A sharing function defines the degree of sharing for each

individual in a population (Goldberg 1989). The degree of sharing for one individual is calculated by summing a set of sharing function values that indicate the distance between it and the other individuals in the population having the same rank. Fig. 5.1 presents the definition of the sharing function for a problem domain with one objective. The individuals that are close in objective space to another individual have a large degree of sharing and those that are far from another individual have a small degree of sharing.

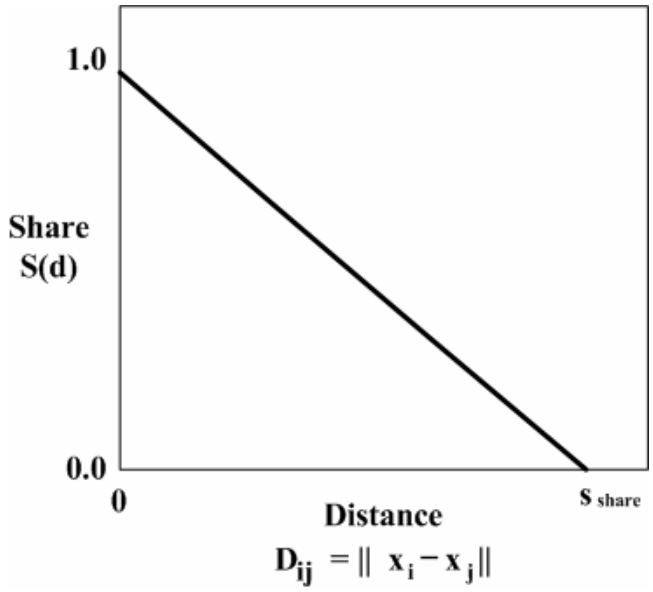


Fig. 5.1. Triangular sharing function (Goldberg 1989)

Equation 5.1 presents the equator used to calculate the shared fitness values for each individual in the population (Goldberg 1989).

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} s(d(x_i, x_j))} \tag{5.1}$$

where

n	= size of population
$f(x_i)$	= fitness value of i^{th} individual
$s(d(x_i,x_j))$	= sharing value based on the distance between i^{th} and j^{th} individual
$f_s(x_i)$	= the fitness value of i^{th} individual considering the degree of sharing
x_i , x_j	= i^{th} and j^{th} individual

Using equation 5.1 to degrade the fitness values of individuals using sharing values, the distribution of individuals across the Pareto front can be better maintained. Having many individuals in the same neighborhood cause them to have the high degree of sharing. Therefore, the fitness values of these individuals considering the degree of sharing will decrease, making them less competitive. Fig. 5.2 presents the general process flow of a MOGA based on Pareto-ranking that implements the sharing function.

5.1.2 Strength Pareto Evolutionary Algorithm

Strength Pareto Evolutionary Algorithm (SPEA) was proposed by Zitzler and Thiele (1999). In SPEA, non-dominated individuals at each generation are archived into an external set that serves as another pool in order to save non-dominated individuals that have been found. For each iteration, the strength values are computed in the external set. The strength of an individual is a similar concept to ranking in that the strength value is in proportion to the number of other individuals that dominate a certain individual. The fitness of each individual in the mating pool was computed according to

the strength of all non-dominated individuals in the external pool that dominate it. The more individuals that dominate it, the lower the fitness assigned.

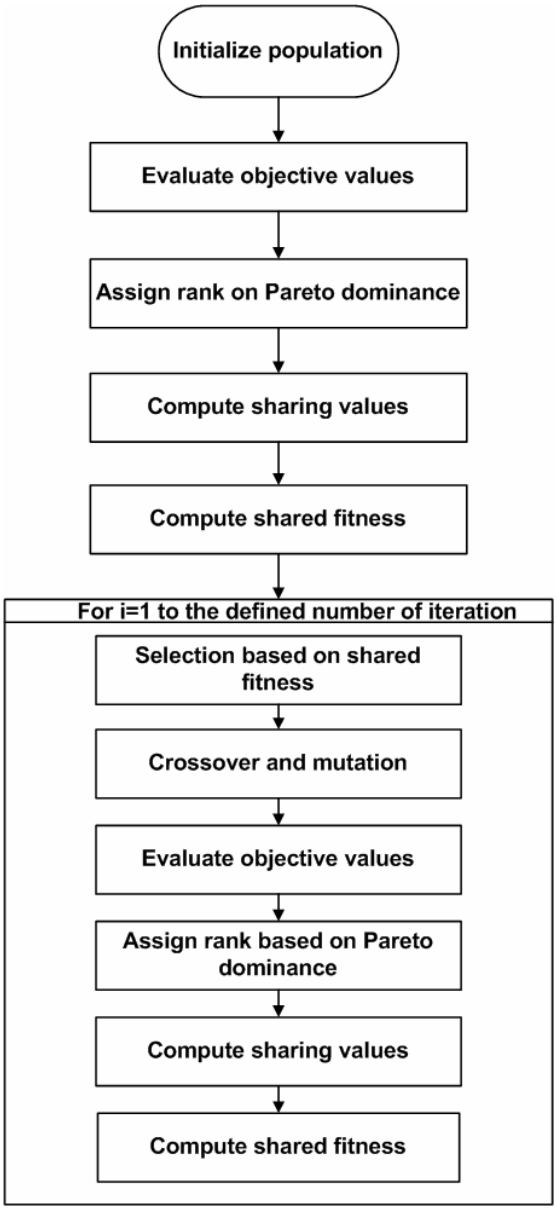


Fig. 5.2. Flow of MOGA based on Pareto-ranking that implements a sharing function

In research performed by Zitzler and Thiele (1999), the solution results of nine functions solved by SPEA were compared with results obtained solving these functions using seven other algorithms, including MOGA using ranking and a sharing function. The SPEA results showed high effectiveness for distributing the non-dominated individuals over the Pareto front (Zitzler and Thiele 1999).

In this Section, the results of trials investigating using a method that incorporates non-dominated Pareto ranking are presented. In this research, the flexibility of the unstructured domain and the design grammar proposed earlier caused several problems in the using computational method as presented in Section 4. To overcome these problems, the Pareto-based ranking, sharing, and strength concepts and strategies proposed previously by other researchers will be reviewed. Then a modified method is proposed to provide an efficient optimization of trusses in the design domain defined in this research.

5.2 MOGA Based on Ranking Using Sharing Function

The multi-objective optimization was performed based on ranking using a sharing function. In all trials, selection was carried out based on a tournament selection scheme. In this method, the highest ranked individual in the tournament set is always selected. In case that an individual has the same rank as the others, then the individual with the highest shared fitness value is selected. The total fitness of each individual, $f_{TOT,j}$, can be computed using the composite fitness function that was formulated in Section 4.

Unlike optimization using the composite fitness function, all the constraints were converted into independent objectives. In equation 5.3, the sharing function values were computed considering two objectives that were to be minimized, which were the truss weight and deflection.

$$f_{Shared,i} = \frac{f_{TOT,i}}{S_i} \tag{5.2}$$

where

$f_{TOT,i}$=total fitness values of i[th] individual.

$$S_i = \sum_{j=1}^{n} C_s - \left(\left| \frac{F_{w,i} - F_{w,j}}{D_{w,MAX}} \right| + \left| \frac{F_{D,i} - F_{D,j}}{D_{D,MAX}} \right| \right) \tag{5.3}$$

where

$F_{w,j}$ = Weight of j[th] individual,
$F_{D,j}$ = Deflection of j[th] individual,
$D_{w,MAX}$ =Maximum difference of weight in population,
$D_{D,MAX}$ =Maximum difference of deflection in population,
C_s =100.

The two objective values for each individual were compared with those of all the other individuals to compute the distances. The summation of the distances for each individual was normalized by the maximum distance, which was determined as the distance between the two individuals that were the farthest away from each other in the population. The string lengths that were found to be efficient in Section 4 were used again for each span length considered. Three span lengths (40 ft., 60 ft., and 80 ft.) were investigated in this Section.

The search domain of the problem considered in this research, however, has a

large infeasible area that includes unstable trusses that are too heavy or have large deflection. This means that the maximum distances used in the sharing function may cause individuals to unnecessarily explore the broad range of infeasible trusses. Therefore, the maximum distances calculated had to be reduced in order to provide an efficient exploration by multiplying the maximum values $D_{w,MAX}$ and $D_{D,MAX}$ by constants that were empirically determined.

$$D_{w,\mathrm{modified\_MAX}} = C_{Dw} D_{w,MAX}\,, \quad D_{D,\mathrm{modified\_MAX}} = C_{DD} D_{D,MAX} \qquad (5.4)$$

where

$0 < C_{Dw}, C_{DD} \le 1,$

$C_{Dw} = 0{,}9,$

$C_{Dw} = 0.6.$

Fig. 5.2 presents the stages defined for the algorithm for MOGA based on ranking and sharing. Computing the rank and the sharing function values required much longer runtimes due to the large number of loops required than the trials performed with the composite function performed in Section 4. Table 5.1 presents the set of GA parameter values used in the trials. Except for the string length, the same parameter values were used for all three span sizes.

Fig. 5.3 presents the result for the 40 ft. span using the defined MOGA approach. The x-axis indicates the average deflection of all nodes in a truss, and the y-axis indicates the total weight of a truss. The trusses defined in the Pareto-optimal front have diverse nodal locations and member sizes. However, most of individuals defined in the Pareto-optimal front have similar truss topology.

Table 5.1. GA parameters in the trials

	40 ft	60 ft	80 ft
Generation	1000	1000	1000
Population	1000	1000	1000
Crossover Probability	0.3	0.3	0.3
Mutation Probability	0.005	0.005	0.005
String Length	300	400	600

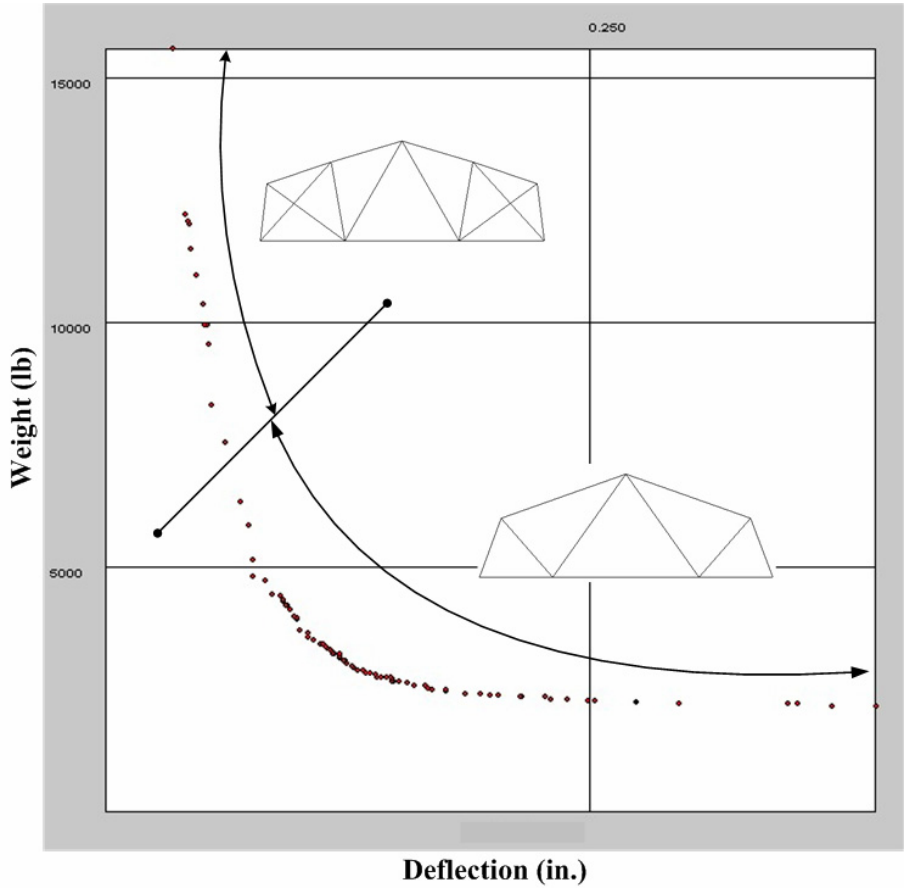


Fig. 5.3. Pareto front for results obtained using MOGA based on ranking.with 40 ft. span

Each representative truss topology in identified in one of two regions as shown in Fig. 5.3. The Pareto-optimal front results showed the monotony of truss topologies, and were composed of only 84 Pareto-optimal trusses even though the population size was 1000. This means that the majority of the trusses in the population did not converge to the Pareto-optimal front because of the limited topologies that were optimal. In addition, changing parameter values and the constants strongly affected the results similar to the sensitivity indicated in the trials performed using composite fitness function in Section 4.

In Fig. 5.4, the Pareto-optimal set results for the trial performed with a 60 ft. span is presented. Compared with the trial for the 40ft. span, a decreased number (49) of individuals were ranked as the highest rank and composed the Pareto-optimal set. In region (1), two different topologies were intermingled along the Pareto front without a strict boundary defined. Similar to the trial with the 40 ft. span, the results obtained were sensitive to the parameter values and constants used in sharing function and the composite fitness function.

The result of the trial performed for the 80 ft. span shows even more monotony in the truss topology defined in Fig. 5.5. The Pareto-optimal front consisted of only one topology with changes of nodal locations and member sizes occurring along the front. Compared with the previous two trials, an even lower number of the highest ranked individuals was found.

From the trials performed with three span sizes, the distribution of individuals was difficult to maintain with the type of sharing function defined for these trials.

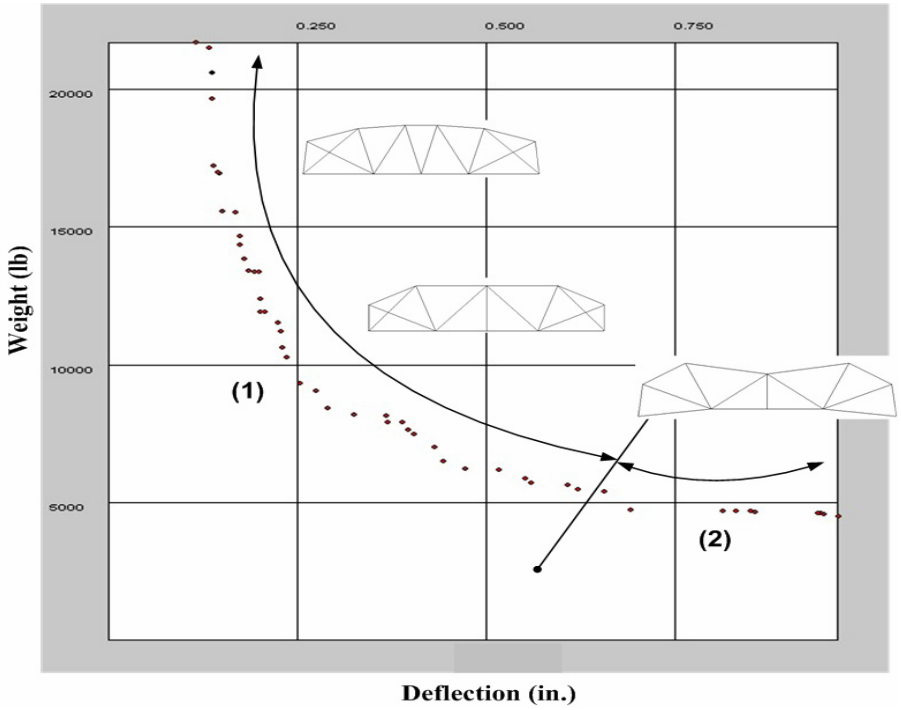


Fig. 5.4. Pareto front from MOGA based on ranking with 60 ft. span

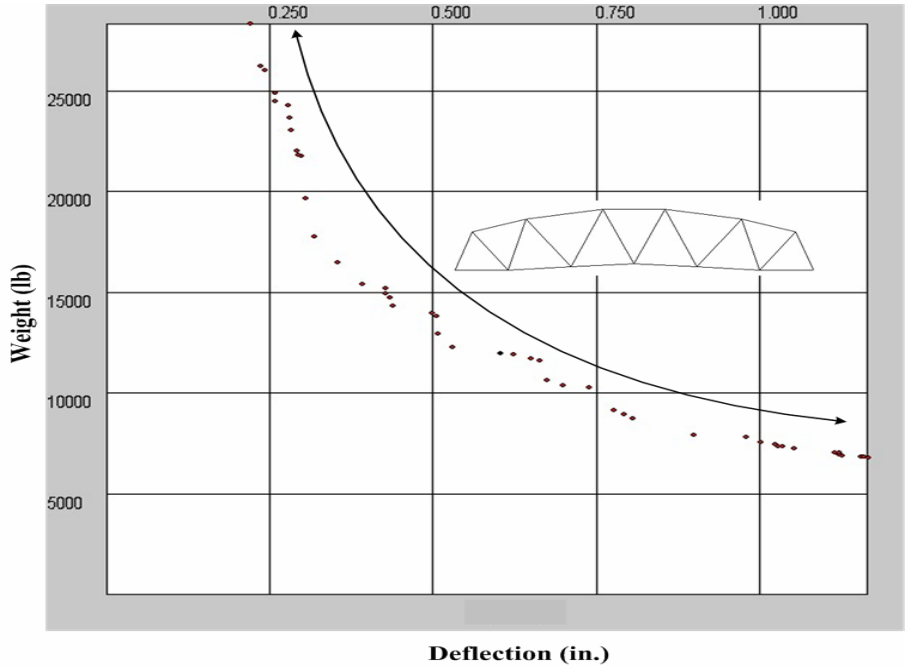


Fig. 5.5. Pareto front from MOGA based on ranking with 80 ft. span

In addition, the small number of the individuals in the Pareto-optimal set (highest ranked individuals) reflects that the individuals were concentrated in local optimum or most of the individuals were in infeasible areas. During the genetic process for the 40 ft. span, the number of feasible individuals was up to 60 percent of total population. However, most of stable trusses in the population showed the lack of diversity in the truss topology and had the same topologies as those presented in the Pareto front. This means that most individuals converged to a local optimum topology. In addition, no boundary between different topologies on the Pareto front was found for the 60 ft. span and only one topology was evolved on the Pareto front for the 80 ft. span. These results indicate that the MOGA with sharing and ranking did not work well for either local or global optimization.

All the problems discussed above were identified to be caused basically from the failure in distributing individuals over the feasible area of the search space. Although the sharing function was applied to the MOGA, the results did not show the creation of an acceptable Pareto-optimal front that was expected to have several design topologies defined.

5.3 Implementing Modified Strength Pareto Evolutionary Algorithm (SPEA)

The ranking scheme used along with the sharing function helps to distribute individuals uniformly. Therefore, the individuals in the population tend to explore the search space superfluously by including infeasible areas. In a problem domain that includes huge infeasible areas, such as the domain examined in this research, however,

the superfluous exploration cannot provide an efficient overall search. In previous trials, it is inferred that the strong force induced by composite fitness function lead most individuals to the local optimum without enough competition with other truss configurations because the superfluous exploration caused only few individuals to be defined in the feasible area.

In Strength Pareto Evolutionary Algorithm (SPEA) proposed by Zitzler and Thiele (1999), the strength is computed in proportion to the number of individuals that dominate a certain individual. Then the fitness of each individual is computed according to the strength of individuals that dominate it. Unlike the sharing function defined previously, in which the niche count is defined in terms of the distance between individuals in objective space, niche count is defined in terms of dominance in SPEA.

For any problem domain that is extremely complicated and has a huge infeasible area, the SPEA is considered an appropriate method to use. The goal of this research is to obtain the near-optimal set, which is expected to contain diverse design alternatives including diverse truss topologies. To achieve the goal, individuals should be well distributed all over the feasible area rather than distributed over all the search space that includes a large infeasible area.

The modified MOGA used in this research is based on the ranking scheme. To compute the niche count, however, the concept of strength was adopted. In addition,other strategies were proposed to help maintain the diverse shapes and topologies of trusses in population and in the Pareto-optimal set.

5.3.1 Formulation of Fitness Function and Penalty Function

The composite fitness function that caused the previous problem with difficulty in converging to a diverse Pareto set due to its strong force is not used in the modified MOGA. However, penalty terms are still used in order to prevent a large number of individuals from being in the infeasible area.

Four penalty terms were defined for the maximum length of each members, the minimum length of each member, perpendicular distance between members and nodes, stability of trusses, and maximum stress for each member as follow:

$$D_{Sh} = \frac{\sum_{i}^{N}(8 - SM_i)}{N} \tag{5.5}$$

where

N = number of a truss members,
SM = length of i^{th} members or perpendicular distance between elements and nodes, (8-SM)<0 then 8-SM=0.

$$P_{DSh} = \frac{DSh}{Max.DSh} \tag{5.6}$$

where

$Max.D_{Sh}$= maximum value of D_{Sh} at a previous iteration.

$$D_{Lo} = \frac{\sum_{i}^{N}(LM_i - 20)}{N} \tag{5.7}$$

where

N= number of a truss members,
LM= length of i^{th} members, (LM-20)<0 then LM-20=0.

$$P_{DSh} = \frac{DLo}{Max.DLo} \tag{5.8}$$

where

$Max.D_{Lo}$= maximum value of D_{Lo} at a previous iteration.

The total penalty imposed on an infeasible individual can be calculated as:

$$P_{TOT} = C_S P_S + C_{DSh} P_{DSh} + C_{DLo} P_{DLo} + C_{st} P_{st} + 1 \tag{5.9}$$

where:

C_s, C_{DSh}, C_{DLo}, C_{ST}: Constants
$C_s = 10$
$C_{DSh} = 100$
$C_{DLo} = 100$
$C_{st} = 50$
P_{st}, P_s: refer to Table 4.3

The penalties concerning constraints defined are combined into a total penalty function (equation 5.9). Unlike previous trials in this research, in which the penalty value was used to decrease the fitness value, the penalty value computed using equation 5.9 concerns only formulating the sharing function, which is used only if individuals in selection are conflicting.

Even though the geometry and the topology of a truss do not change, the two objectives, concerning truss deflection and weight can vary corresponding to the alterations of member sizes. To obtain diverse topologies and geometries, the distribution of trusses related to the topology and geometry should be considered as how well the two objectives are searched. Because the variables related to the geometry and topology of a truss are not objectives, the niche count for them cannot be defined in terms of dominance. Instead, a sharing function is used to compute the niche count for

maintaining the diversity of truss topologies and geometries.

$$F_{share,i} = \sum_{j=1}^{N} \left[C_C - \left(\left| \frac{EN_i - EN_j}{MaxEN} \right| + \left| \frac{NN_i - NN_j}{MaxNN} \right| + \left| \frac{LE_i - LE_j}{MaxLE} \right| \right) \right] \qquad (5.10)$$

where

EN_i, EN_i	= Number of elements in the ith and jth truss
NN_i, NN_i	= Number of nodes in the ith and jth truss
LE_i, LE_i	= Total length of members in the ith and jth truss
MaxEN	= Maximum value of EN in previous iteration
MaxNN	= Maximum value of NN in previous iteration
MaxLE	= Maximum value of LE in previous iteration
N	= Number of trusses in a iteration
Cc	= Constant, which was set equal to 100

Finally the new fitness function for the MOGA was formulated as follows.

$$F_{TOT} = \frac{C_{TOT}}{F_{share} P_{TOT} F_{strength}} \qquad (5.11)$$

where

$F_{strength}$ = 1+ the number of individuals that it dominates,
C_{TOT} = Constant, which was set equal to 1.

In equation 5.11, the composite penalty value provides feasible individuals with more opportunities to be selected. The sharing function defined in terms of the geometry and topology variations enables fewer trusses with the same topologies and geometries to be selected. In addition, $F_{strength}$, which is based on the concept of strength, is defined to compute the niche count for how well the design objectives are satisfied. The fitness function was formulated to focus on the distribution of diverse individuals in the feasible areas rather than all over the search space.

In SPEA (strength Pareto evolutionary algorithm) proposed by Zitzler and Thiele (1999), the Strength is computed only for the individuals in the external set and

used to compute the fitness and niche count of each individual in the mating pool. In this modified algorithm, however, the Strength, $F_{strength}$, is computed for all the individuals in the mating pool, external pool, and the backup pool and used as a measure to determine a density of a local area only if the individuals in selection conflict.

5.3.2 Performance of the Modified Algorithm Using SPEA and MOGA Based on Ranking

The new algorithm developed is based on both SPEA (Zitzler and Thiele 1999) and MOGA based on Pareto-ranking. Tournament selection is performed based on the rank of an individual as discussed previously in section 5.2. The case that two individuals have the same rank, the individual that has the higher fitness that is computed with equation 5.11 is selected.

To assist in maintaining the diverse topologies of trusses, another pool that was named the backup pool was added to the modified algorithm. In the backup pool, a representative of each truss that has the similar shape and topology is saved. This means that one representative is selected among all the trusses that have the same number of nodes, elements, and a similar total length of members. If the difference of the total length between two trusses was within 20 percent, the two trusses were assumed to have the same total length. This representative is saved to backup pool. To increase the probability of selection for the individuals in backup pool, the representatives are duplicated several times and saved in the backup pool.

Fig. 5.6 presents the steps defined in the modified MOGA proposed in this

research. Selection is performed in the multi-set population (mating pool + external set + backup pool) until the mating pool is filled for the next generation. After crossover and mutation, the individuals in mating pool and external set are sorted by their rank and their fitness. The individuals in the previous external set are replaced with the newly ranked individuals. In addition, the representatives of new topologies that are found in the mating pool at each generation are saved in the backup pool. For the selection, the rank and fitness are computed for trusses all the three pools.

Three truss spans (40 ft., 60 ft., and 80 ft. span) were investigated to determine the performance of the modified MOGA. The objectives, constraints, and prescribed conditions, such as selected members and maximum height of the design domain were the same as those used in previous trials, except for the constraints that imposed limit member length (equations 5.5~5.8). Each constraint to limit member length was imposed as a penalty function in terms of maximum and minimum distances.

The GA parameters used were the same as stated previously (table 5.1), and the size of backup pool and external set was specified as the same as those of the mating pool, was 500.

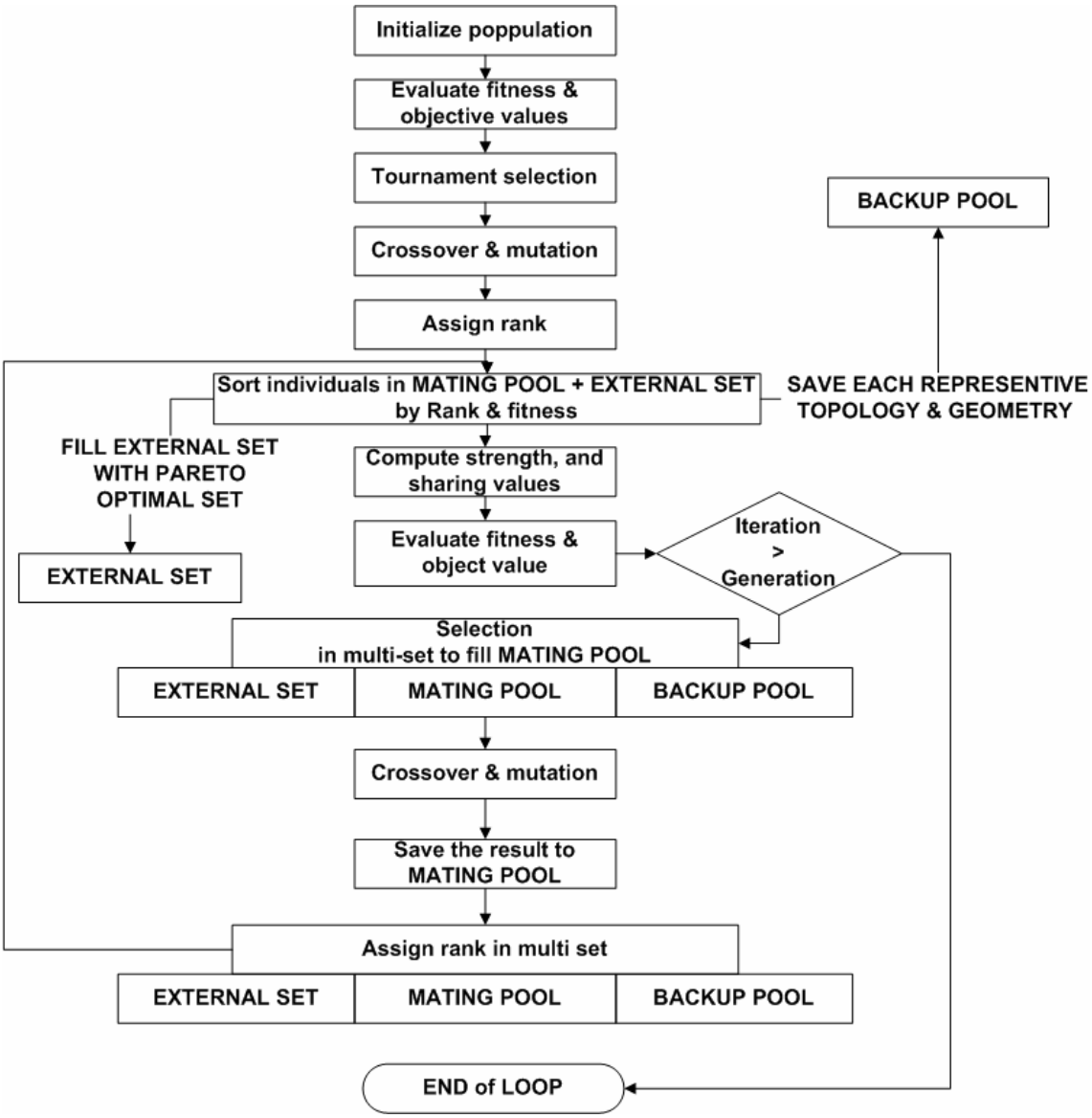


Fig. 5.6. The flow of modified MOGA incorporating mating, external, and backup pools

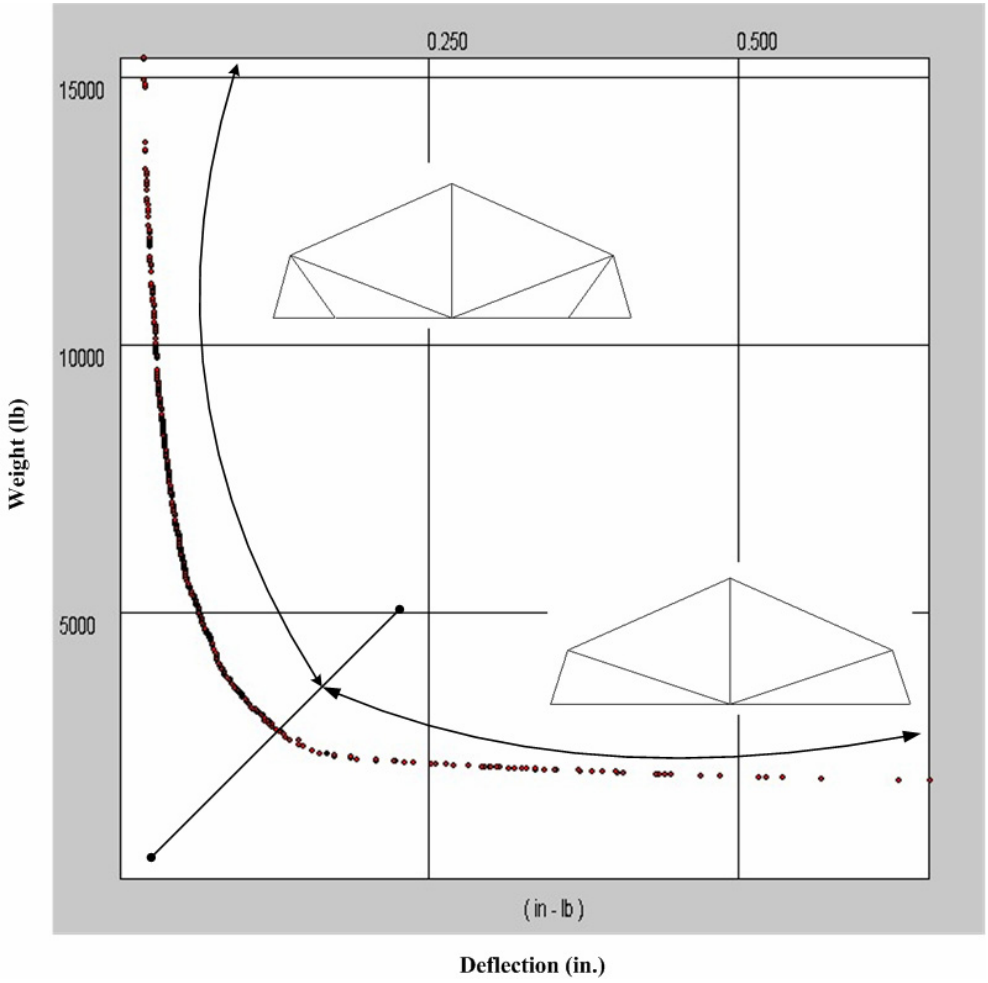


Fig. 5.7. Pareto-optimal front for 40 ft. span obtained using modified MOGA

Fig. 5.7 presents the results obtained from the trial performed using the modified MOGA for the 40 ft. span. For this trial, a 500 sized array was allocated to each pool. The Pareto-optimal front is composed of two topologies with different nodal locations and member sizes. Compared with the previous results that were performed using MOGA based on ranking using sharing function only, the curve obtained with the modified MOGA was smoother, which could indicate that the trusses are locally well

optimized. The Pareto-optimal front obtained for the 60 ft. span was similar to those obtained for the 40 ft. span. Two different topologies of trusses were shown in the defined Pareto-optimal front as shown in Fig. 5.8.

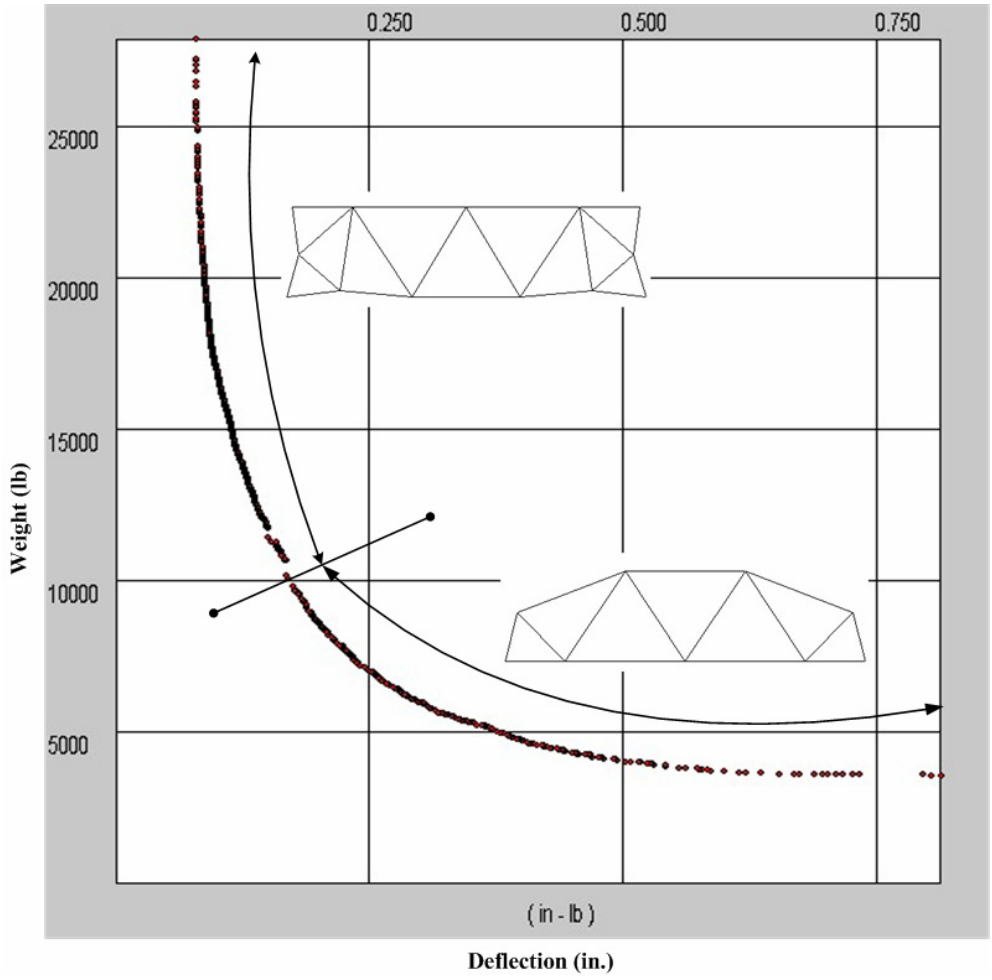


Fig. 5.8. Pareto-optimal front for 60 ft. span obtained using modified MOGA

This type of result presented reasonable changes in truss topology and geometry according to changes in the weight and deflection objectives. According to the

increment of weight, the member sizes and the number of members in a truss increased as well as the overall height of the truss. In addition, the results obtained in previous trials without the modifications showed fewer individuals in the Pareto-optimal front for trials with longer span length. However, the design domain expanded due to the increase in span size should contain more design alternatives. Moreover, a larger number of members in a truss should enable more combinations of member sizes to be defined in the optimal set.

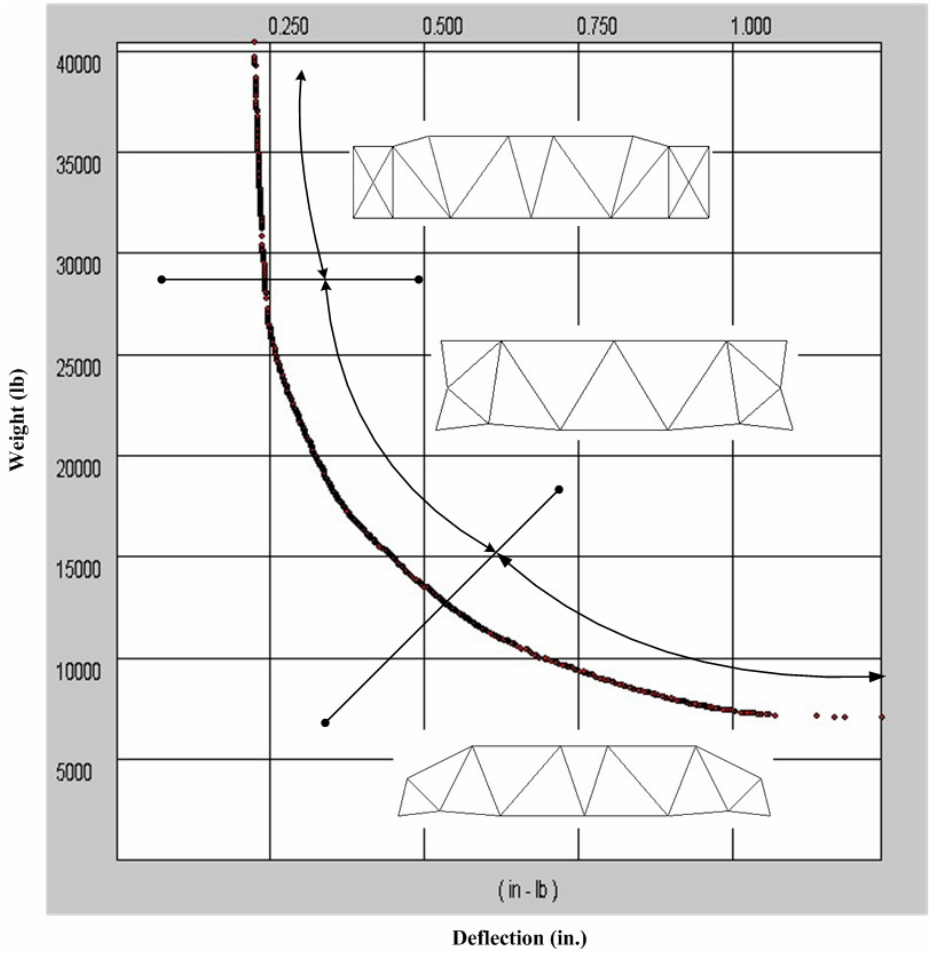


Fig. 5.9. Pareto-optimal front for 80 ft. span obtained using modified MOGA

The results from the modified MOGA proposed showed a larger number of individuals in Pareto-optimal front for longer span lengths. This improvement means that the modified algorithm and strategies enable individuals to effectively explore the search domain for feasible trusses. In Fig. 5.10, the results obtained from the modified MOGA are compared with the previous results obtained using just the MOGA without modifications.

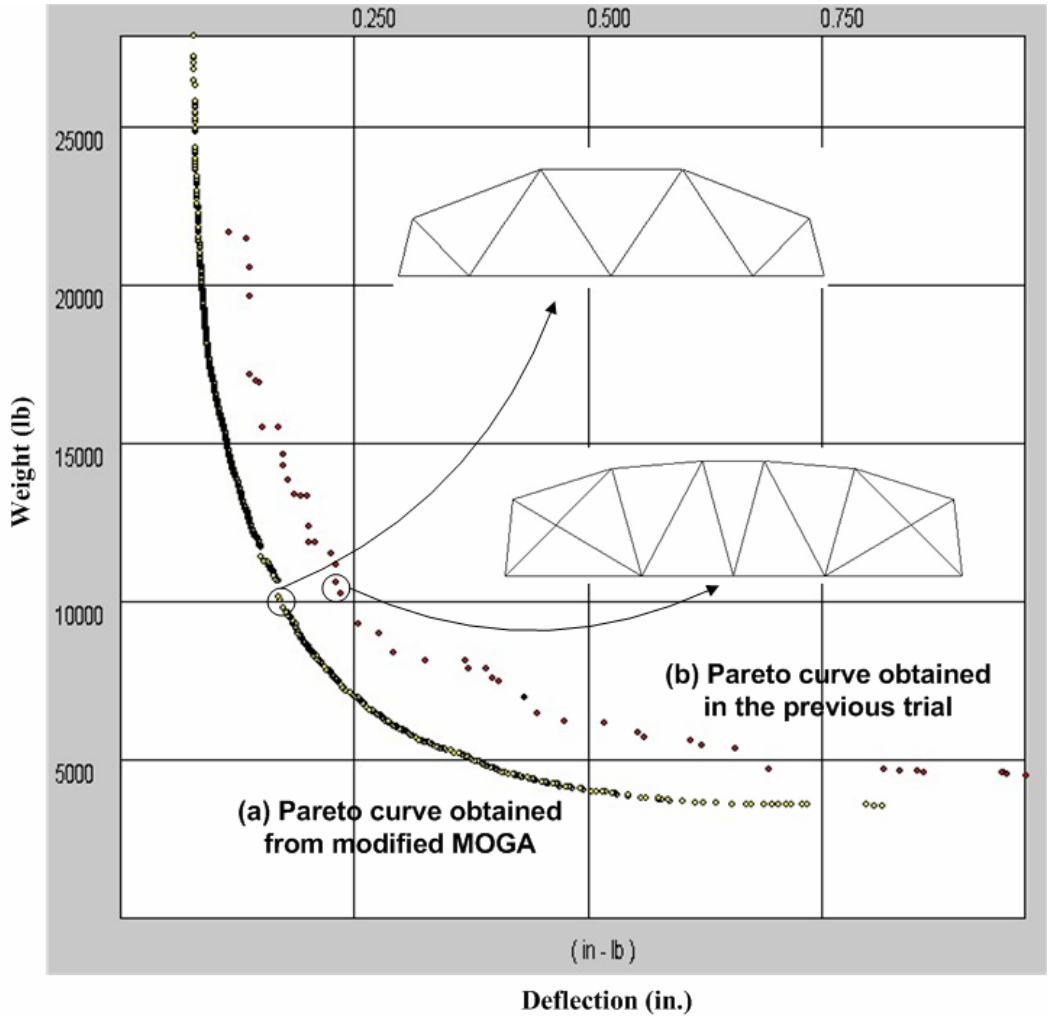


Fig. 5.10. Comparison of the result with the previous result in 60 ft. span

The comparison shows remarkable improvement in the result obtained. As well as increasing the number of individuals in the Pareto front, improved objective values were obtained. In addition, the Pareto-optimal individuals were well-distributed over the front.

The topologies of two trusses that are close to the 1000 lb. are presented in Fig. 5.10. The Pareto front (a) indicates a clear boundary between the different topologies of trusses, whereas the different topologies are placed sporadically along the Pareto front (b). Based on these results, the modified MOGA is more effective at least in the aspect of local optimization. As seen in Fig. 5.11, however, a diverse set of truss topologies were saved in the backup pool.

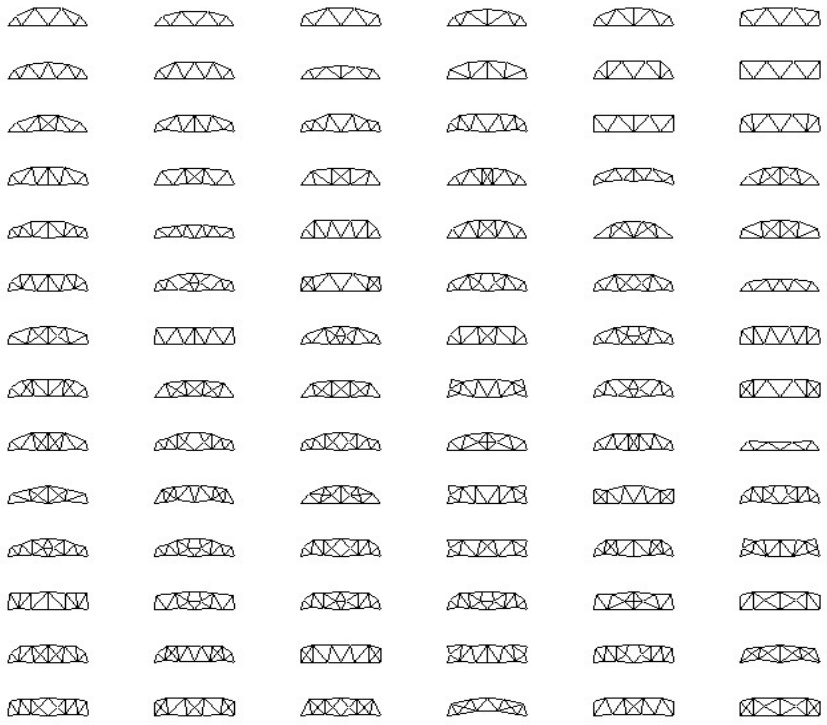


Fig. 5.11. Stable trusses in the backup pool for 60 ft. span

Some of these trusses in the pool have many members which cause low deflection and large weight, which could be presented in the large weight region of Pareto front. In addition, the Pareto front for the 80 ft. span shown in Fig. 5.10, indicates that all the three types of topologies have relatively many members even though a simpler topology could present a non-dominated individual in a large deflection region on the Pareto-optimal front.

From the behavior observed, it is calculated that the locally optimized individuals might have better objective values than others, even though the topologies were not better than others. In this case, these locally optimized individuals would prevail in selection. Consequently other possibly better topologies that were not locally optimized would be removed from population due to poor objective values and would rarely have a chance to be locally optimized and exist on the Pareto front.

5.4 Investigation of the Modified MOGA Using Sub-Processes

The Pareto front obtained using the modified MOGA contains locally well-optimized trusses but not global optimal solutions. The modification of the proposed modified MOGA strategy and algorithm was essential in order to provide the premature individuals with a chance to be locally optimized.

To provide a more effective global exploration, the MOGA algorithm using sub-processes and a restart strategy was investigated. From previously performed research, multi-step processing was investigated and proved to be very effective with respect to both quality of results and computational expense.

In general, the initial seeds of GAs completely depend on random values. In several mathematical programming methods, however, the initial seeds are determined using the result from a previous computation by restart. In the research performed by Rajan (1995), the use of restart was illustrated. As concluded in their research, it was stated that the result that was produced using the previous result by restart was close to the previous result (Rajan 1995).

5.4.1 Test of Sub-Process and Restart Strategy in the Unstructured Domain

In this research, the use of restart was considered to solve the problem that was found in previous trials. The restart strategy would help the premature individuals have a chance to be developed into more efficient truss topology, geometry, and size. Unlike the structured problem domain, in which all the genes are used to construct the phenotype, the restart in the unstructured problem domain would develop the useful redundant genes, which are not currently used to assemble the truss. The modified and improved redundant gene information may assist in obtaining better topologies and geometries as well as the member sizes in later generations.

Before detailing the restart strategy and algorithm, a trial was performed to investigate the effectiveness of the restart in the unstructured design domain. As illustrated in the previous trials, several truss designs are saved in the backup pool. For the investigation, one of the trusses from the backup pool of first process was selected. The initial mating pool of the second process was filled with duplicate copies of the selected truss. In this investigation, the effect of the priorities that are encoded in

individual strings and used to assemble trusses was also investigated.

Fig. 5.12 presents part of the initial seeds for a second process. The initial mating pool was filled with duplicated copies of one truss that was selected in the backup pool for the first process.

Fig. 5.12. Portion of initial seed pool, which is composed of duplicated trusses selected from the backup pool of the first process

In Fig. 5.13, the Pareto front resulting from running the modified MOGA for the second process is shown. The diversity of the truss topology and geometry obtained is improved compared with those obtained from the first process.

From the behavior observed in the results, the restart using the previous results as seeds enabled the individuals that did not have a chance to be improved due to the locally converged individual to have the opportunity to be optimized. Therefore, the algorithm using restart strategy was considered to be effective to solve the problem that was raised using the previous modified MOGA algorithm. The same investigation was also performed with 4 priority values. As seen in Fig. 5.14, the two Pareto fronts

obtained from the two processes exactly coincided.

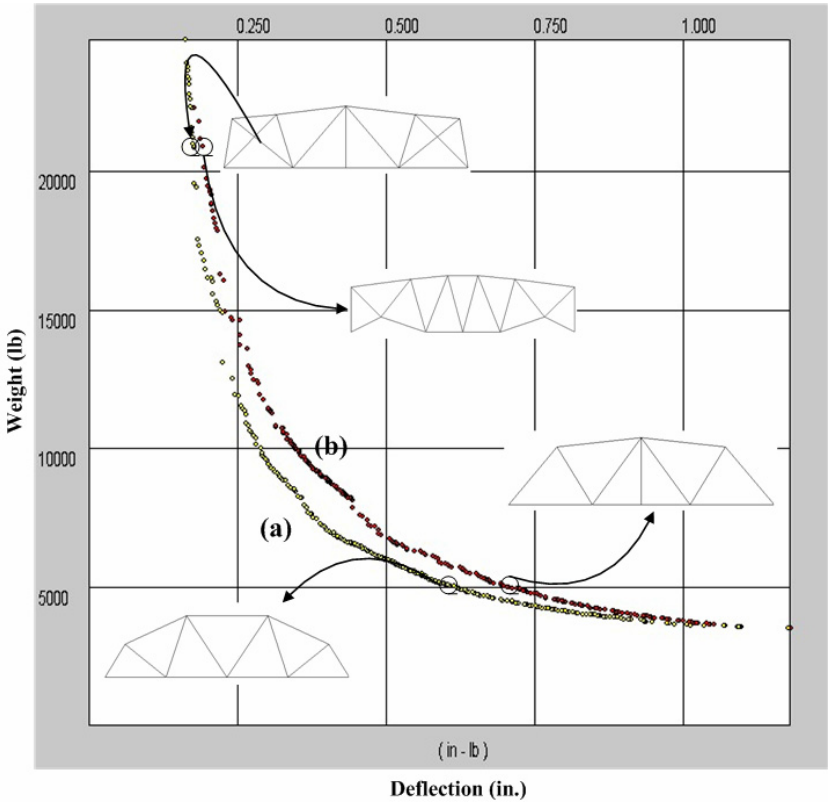


Fig. 5.13. Pareto fronts from (a) second process; (b) the first process with a priority value of 2

In addition, the second process generated new topologies and shapes that were not found in first process. The priority used in this research influences which nodal information was used to form a truss in case the generated nodes conflicted. Among the conflicting nodal information, only one is available to form a truss, and the others remain in the strings as redundant information. From the result obtained, it was concluded that a large allocation of priority caused the strings to contain lots of redundant information.

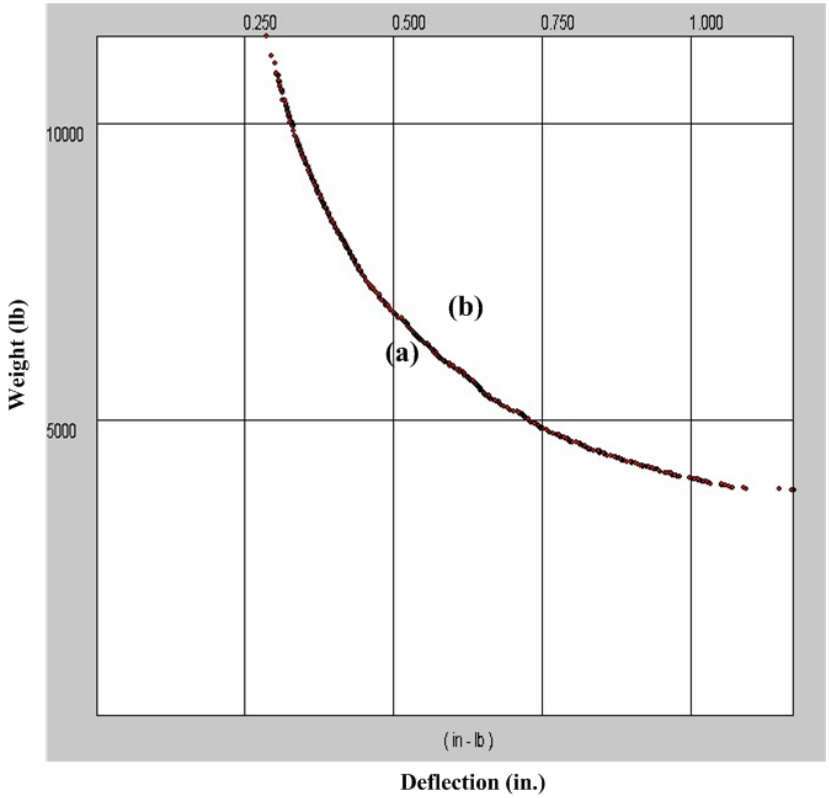


Fig. 5.14. Pareto fronts from the second process (a) and first process (b) with 4 priority value

Therefore, the excessive redundant genes might include a large amount of information about the individuals in the first process including the one that was dominant in the first process. Consequently, the second process might take a similar phase to the first process. More experiments are needed to find the clear reason of this result.

5.4.2 Implementing Restart Strategy Using Sub-Process Algorithm

According to the above results, the restart strategy was applied to the previously proposed modified MOGA algorithm with two priority values. To apply the restart strategy to the previous algorithm, a new architecture was proposed. Fig. 5.15 and 5.16 illustrate the architecture of the genetic process proposed in this research.

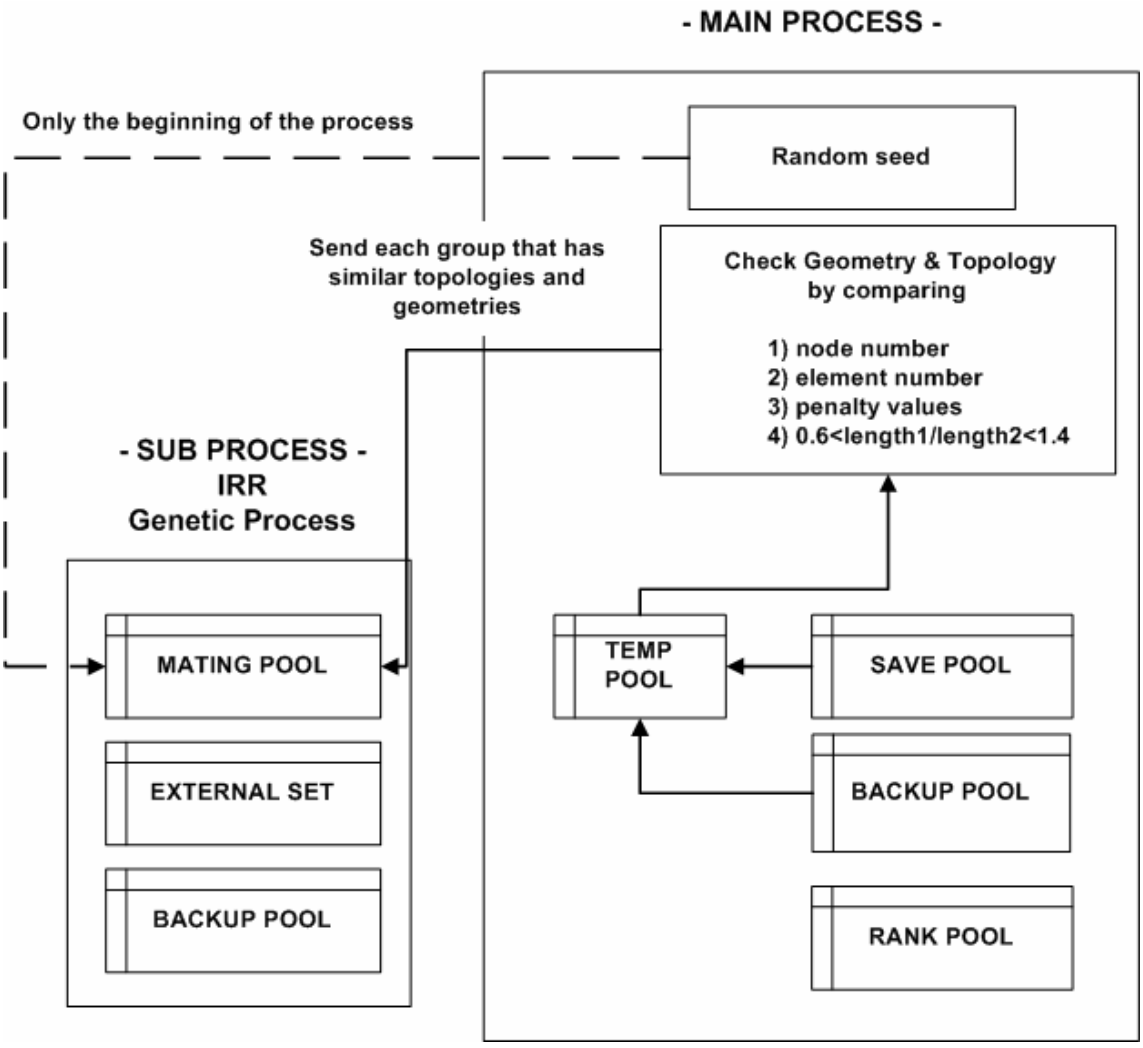


Fig. 5.15. Initialize mating pool in sub process

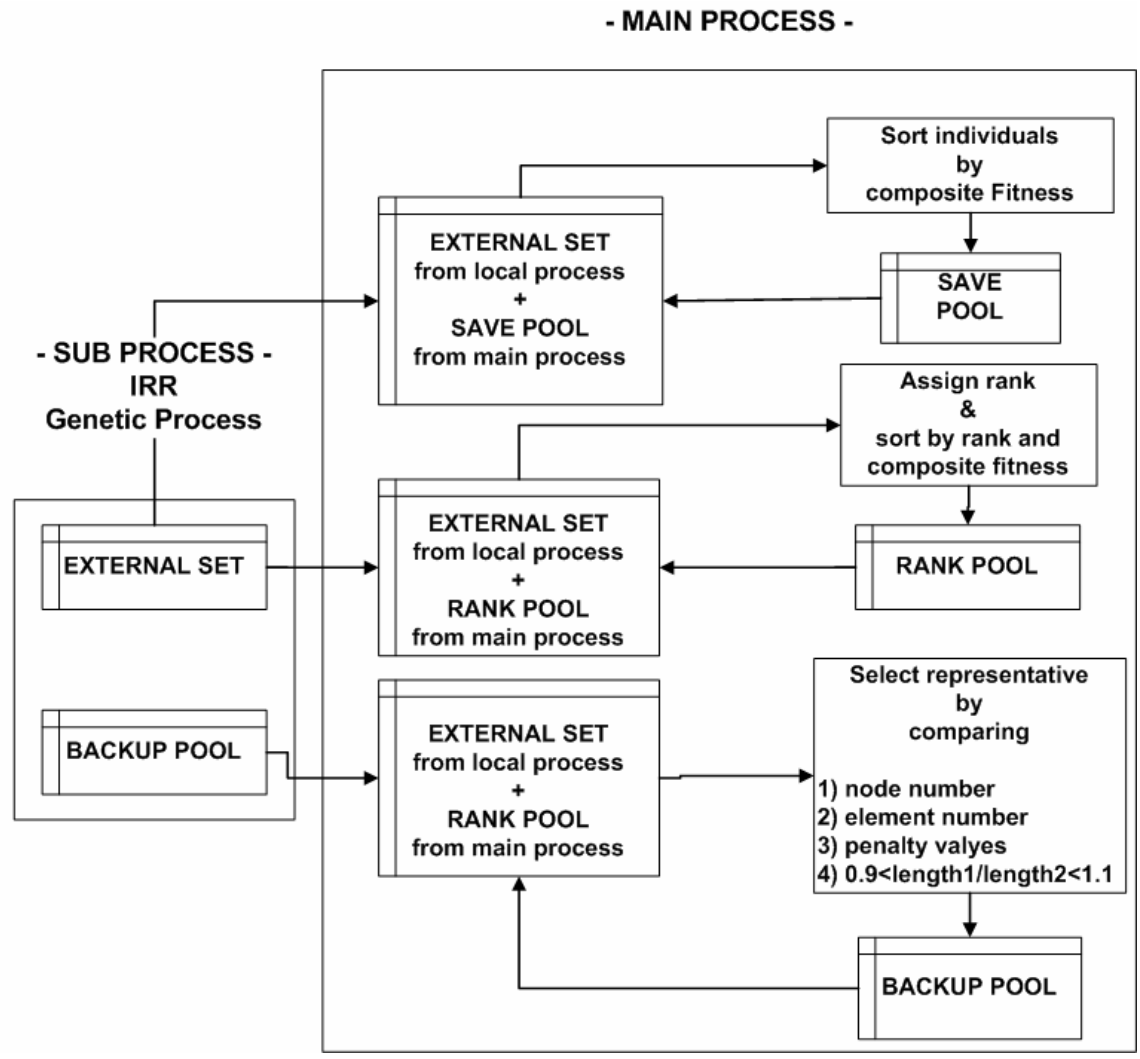


Fig. 5.16. Transfer of the results of the genetic process in sub-process to main process

The algorithm is composed of two processes, which are called the main and sub process. The task of main process is to assign a rank to the individuals, classify the individuals according to the topology and shape, and save representatives of each topology and the higher ranked individuals. The IRR genetic process is performed in the

sub-process. Briefly, the genetic process is preceded in the sub-process, and the main process handles all the process for the seeds and the results of the sub-process.

In case that the pools (rank, backup, and save pool) in the main process are full of individuals, it is possible that higher ranked or fitter individuals cannot be stored to the pools because of the limitation of memory allocation. To prevent the higher and fitter individuals from being removed due to the shortage of allocated memory, the individuals in the pools of main process were sorted every sub-iteration by rank and fitness, as defined in the Section 4. In addition, the individuals in the sub-process were subject to sorting. However, the weight values were applied to the composite fitness function in sub-pool. Without the weight values, the individuals that have the same rank would be sorted by fitness that was computed by the fixed fitness function.

The fixed fitness function causes individuals at certain regions on a Pareto curve to always have the highest fitness. In case that the local pools are full of individuals, the individuals are sorted based on the region they occupy, and the results from the sub-process may concentrate on the region though the seeds of the sub-processes are various. Consequently, the large portion of the Pareto front cannot be obtained due to the shortage of allocated memory in sub-process. Using weight values helps overcome the shortage of assigned memory and enables individuals that are located in diverse regions to be obtained even though the regions may not be continuous due to the discrete weight values. The following section illustrates the steps of the proposed algorithm using sub-process:

1. *Generate the initial individuals randomly:* ***main process***

2. *Transfer the initial individuals to mating pool of sub-process*
3. *Genetic process (Fig. 5.6):* ***sub-process***
4. *Transfer the result of step 3 to main process*

   4.1 Transfer the individuals in the backup pool of sub-process to backup pool in main process

   4.2 Transfer the individuals in the external set of sub-process to save pool of main process

   4.3 Transfer the individuals in the external set of sub-process to rank pool of main process

5. *Generate seeds for next sub-process:* ***main process***

   5.1 Shuffle individuals of save pool and backup pool and put them into temporary pool

   5.2 Classify the individuals in the temporary pool according to the topology and geometry and generate ***n-groups****, in which the trusses have similar topology and shape*

   Every individual assign to the same group should satisfy the following conditions:

   (1) The number of nodes in each truss should be the same as those in the other trusses in a group

   (2) The number of members in each truss should be the same as those in the other trusses in a group

$$(3)\ \ 0.9 < \frac{total \cdot length \cdot of \cdot members \cdot in \cdot truss \cdot A}{total \cdot length \cdot of \cdot members \cdot in \cdot truss \cdot B} < 1.1 \qquad (5.12)$$

5.3 Transfer the individuals in each group to mating pool in each sub-process as seeds: ***n sub-process created****.*

6. *Genetic process preceded with the seeds transferred from main process:* ***sub-process***

   6.1 During sub-process, the pools are sorted by rank and fitness.

$$F_{TOT} = \frac{C_T - (W_W F_W + W_D F_D)}{(C_S P_S + C_{DS} P_{DS} + C_{st} P_{st} + 1)} \qquad (5.13)$$

where

W_W, W_D: weight values for each objectives,
which are assigned randomly as 1,1/3,1/5,1/10,1/100,1/1000.

The weight values are randomly determined and changed according to each iteration of main process

7. *Transfer the results of sub-process to main process*

   7.1 Transfer the individuals in the backup pool of sub-process to backup pool in main process

   (1) Put the individuals in the backup pool of sub-process and main-process to a temporary pool.

   (2) Select representatives of each topology of truss (the representative should have the highest fitness among the trusses that have similar topologies and geometries)

(3) Save the representatives to the backup pool of the main process

7.2 Transfer the individuals in the external set of sub-process to rank pool of main process

(1) Put the individuals in external set of sub-process and in rank pool of main-process to a temporary pool.

(2) Assign rank to individuals in the temporary pool.

(3) Sort the individuals by rank and fitness

(4) Save the individuals to rank pool of main process

7.3 Transfer the individuals in the external set of sub-process to rank pool of main process

(1) Put the individuals in external set of sub-process and in save pool of main-process to a temporary pool

(2) Sort the individuals in a temporary pool by the fitness

(3) Save the individuals to save pool of main pool

8. Go to step 5 until the main iteration < the defined generation size

The number of groups that was created in the main process defines the number of sub-processes performed in each main iteration. In addition, the weight values are included in the fitness values in the sub-process, yet not in the main process.

Trusses with 40 ft., 60 ft., and 80 ft. spans were subject to investigations concerning the performance of new MOGA algorithm defined, and the results obtained

were compared with those from the trials performed with the modified MOGA without sub-process. Unlike previous trials, the maximum deflection of a certain node in a truss represented the deflection of the truss, instead of the average deflection. Table 5.2 presents the GA parameters used and the pool sizes of the main and sub process.

Table 5.2. GA parameters and size of pools for the trials

Main Process		
Iteration	100	
Rank Pool Size	3000	
Backup Pool Size	300	
Save Pool Size	6300	**String Length**
TMP (temporary) Pool Size	3300	40 ft. : 300
Sub Process		60 ft. : 400
Generation	300	80 ft. : 600
Mating Pool Size	300	
External Set Size	300	
Backup Pool Size	300	
Crossover Probability	0.3	
Mutation Probability	0.005	

Fig. 5.17 presents the Pareto front for the 40 ft. span truss. The Pareto front is composed of three different topologies. Compared with the previous result that was

produced without the sub process and the restart strategy, individuals with a wide range of objective values were obtained. In addition, more diverse topologies were saved in the main backup pool. The three trusses presented in Fig. 5.17 indicate the dominant topology of each region.

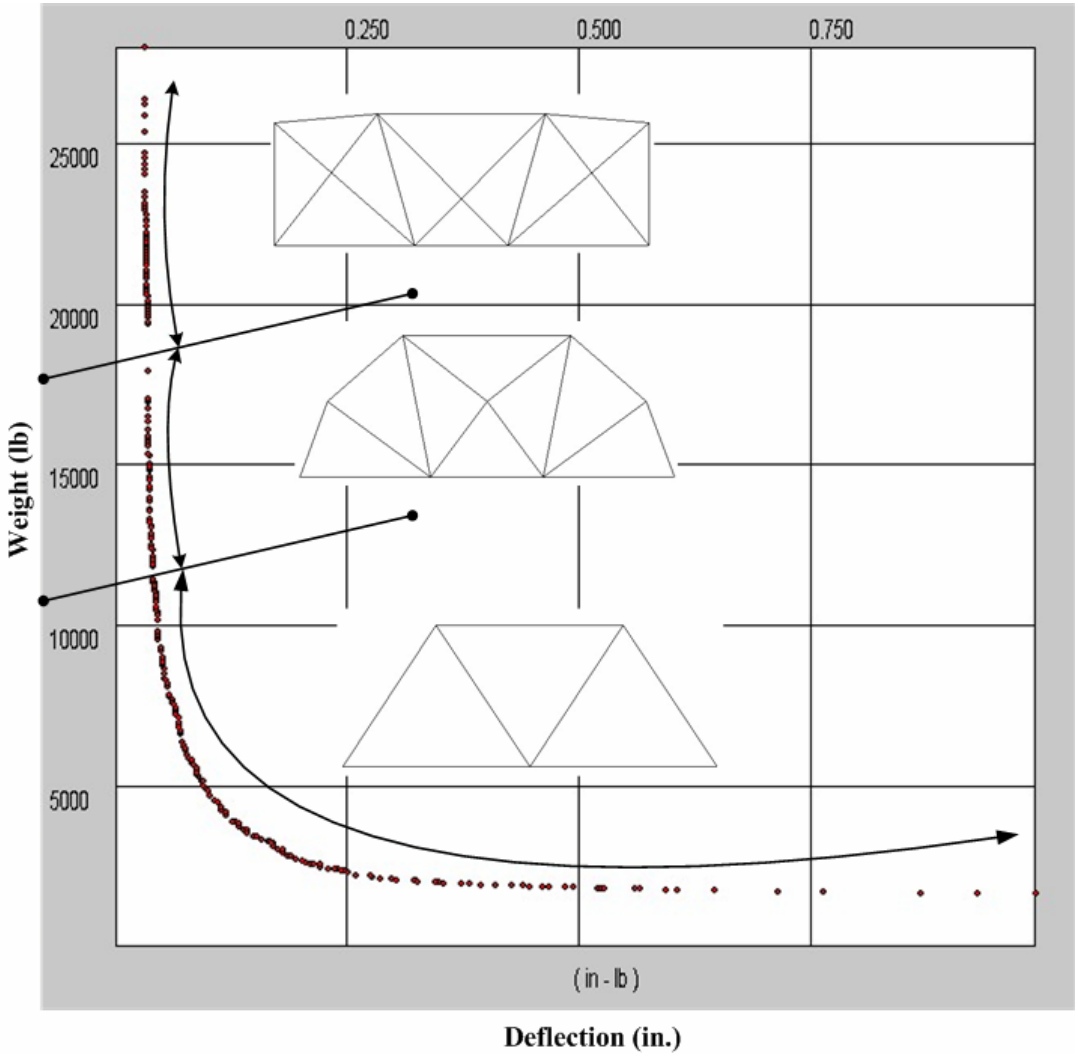


Fig. 5.17. The Pareto front obtained using restart strategy (40 ft.)

Within each region as the objective values changed, movement in the nodal locations in the trusses defined along with changes in member sizes was observed. The increase in the height of trusses caused the deflection to be decreased. When the height of a truss reached the maximum height with close to maximum member sizes, the truss topology changed into one with more members. As presented in Fig. 5.18 and Fig. 5.19, the results obtained for trials investigating the 60 ft. and the 80 ft. span lengths showed very

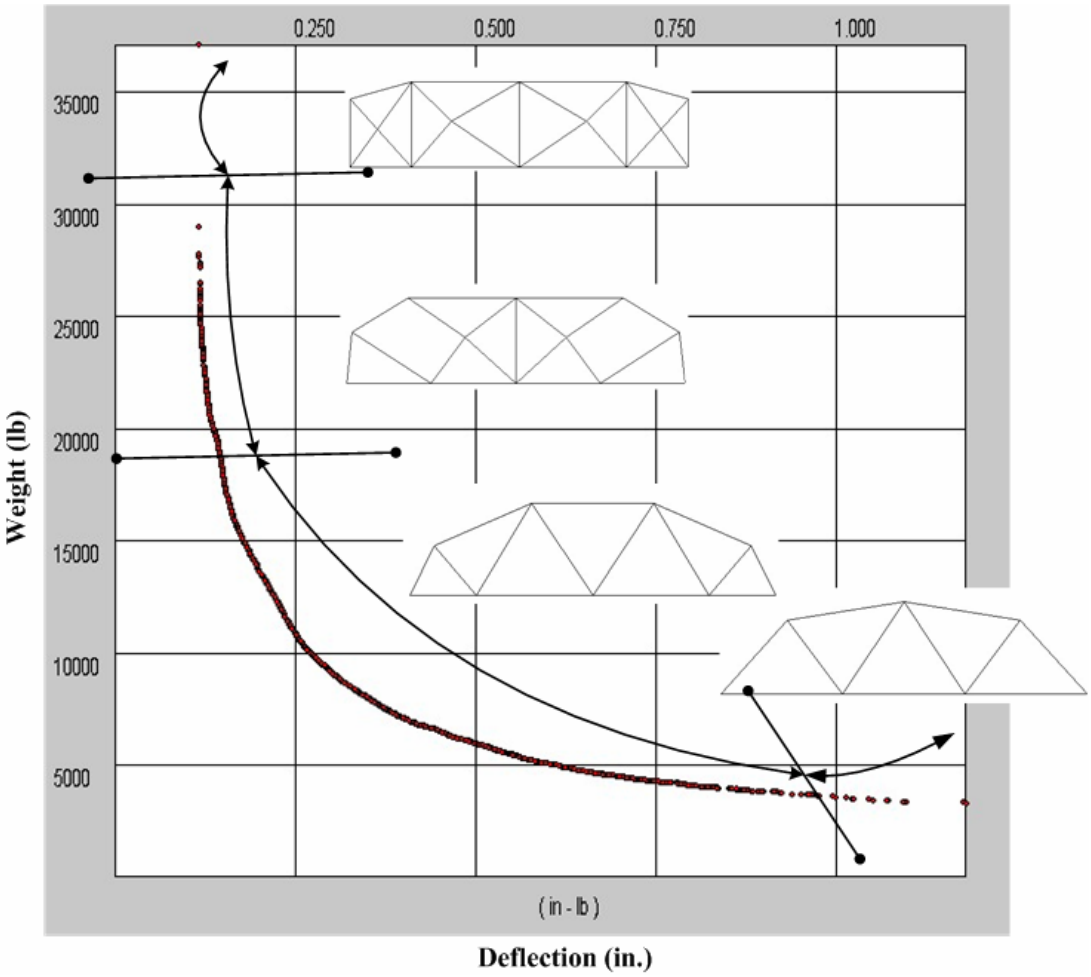


Fig. 5.18. The Pareto front obtained using restart strategy (60 ft.)

similar behavior to the results obtained for the 40 ft. span length according to changes of the deflection and weight objectives. During the MOGA process investigating the 80 ft. span, a large number of trusses were generated. Consequently more groups were obtained. As mentioned in the illustration of the algorithm, each group identified is allocated to a sub-process that consists of one independent IRR genetic process. Accordingly, the runtime of the overall MOGA process increased with the longer span

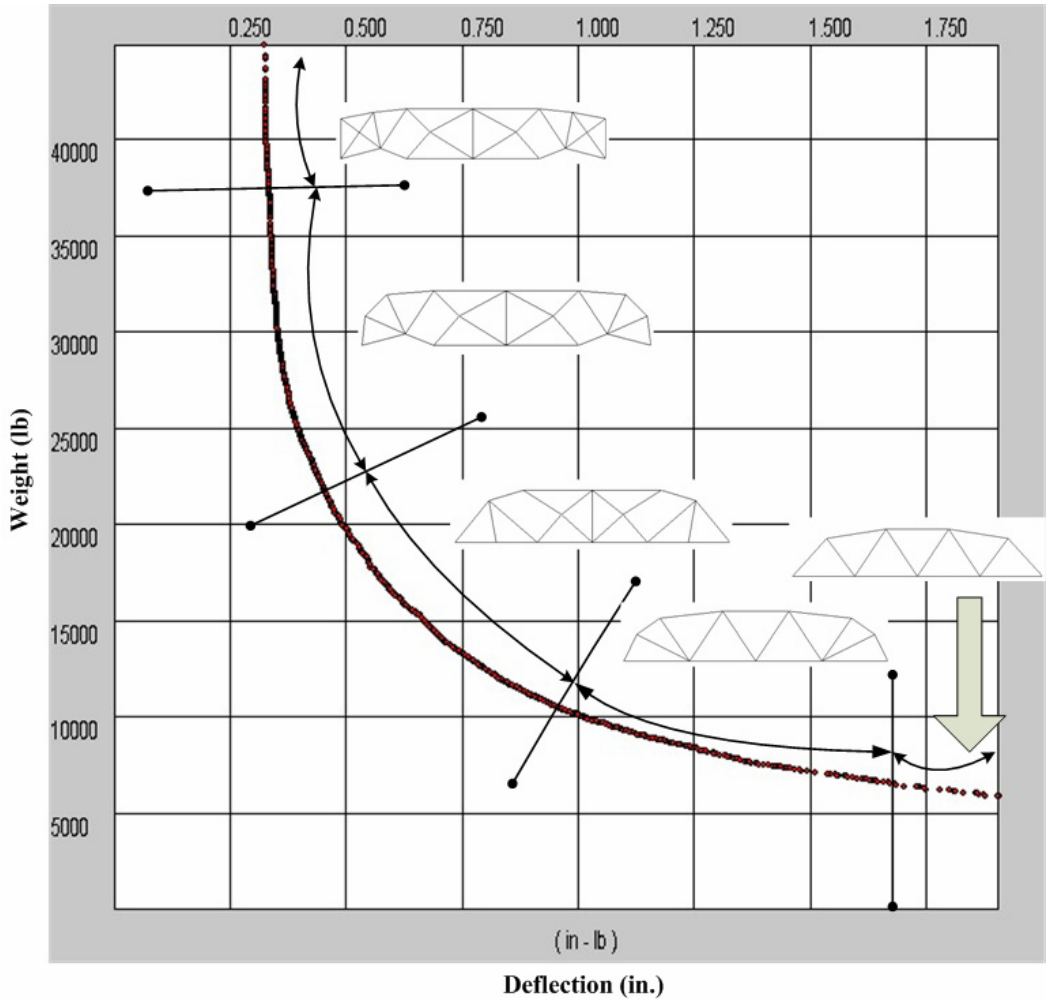


Fig. 5.19. The Pareto front obtained using restart strategy (80 ft.)

length, although the benefit is that more design alternatives are generated. As shown in Fig. 5.19, more design alternatives are presented on the Pareto front. The Pareto front is composed of 1431 trusses and each truss has different nodal locations or a different combination of member sizes.

In Fig. 5.20, the results obtained using the MOGA with restart strategy is compared with those obtained from the previous algorithm that was implemented

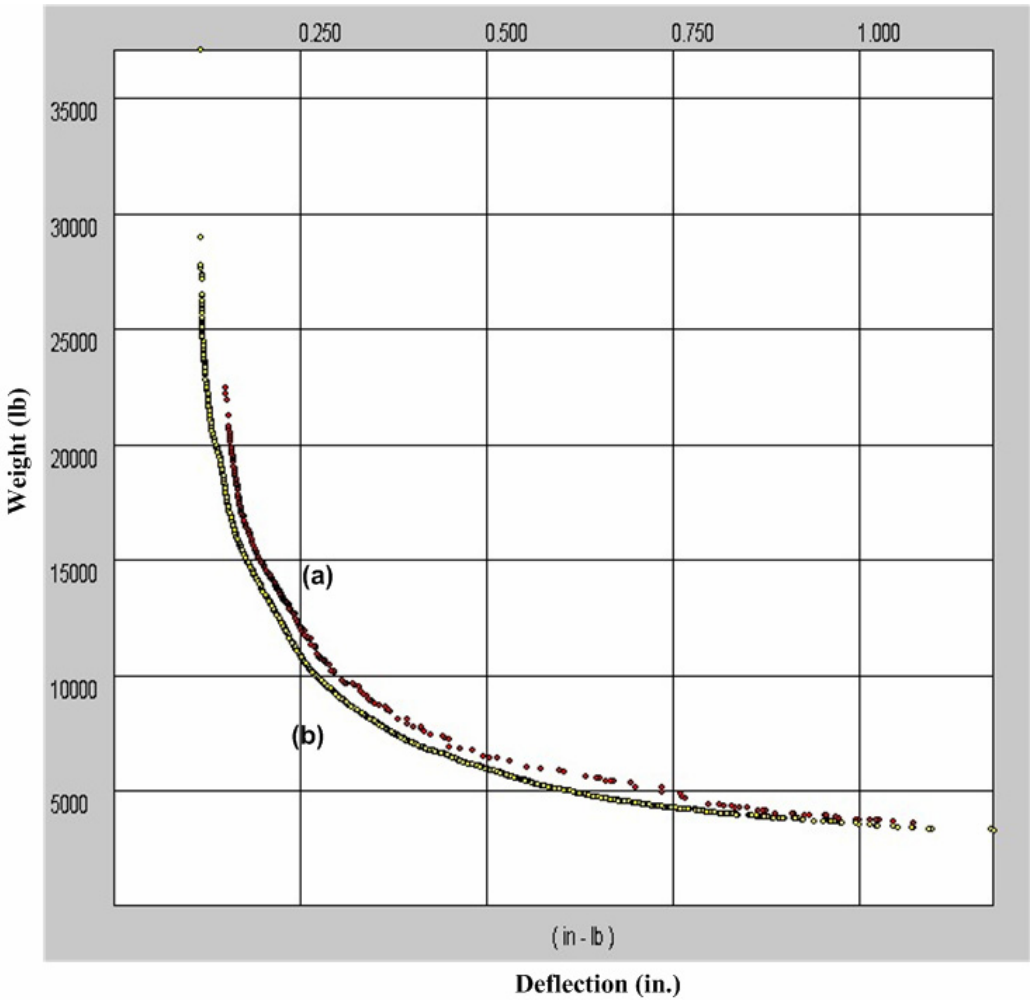


Fig. 5.20. Comparison with the previous algorithm (a) previous results; (b) results obtained using MOGA with restart and sub-process

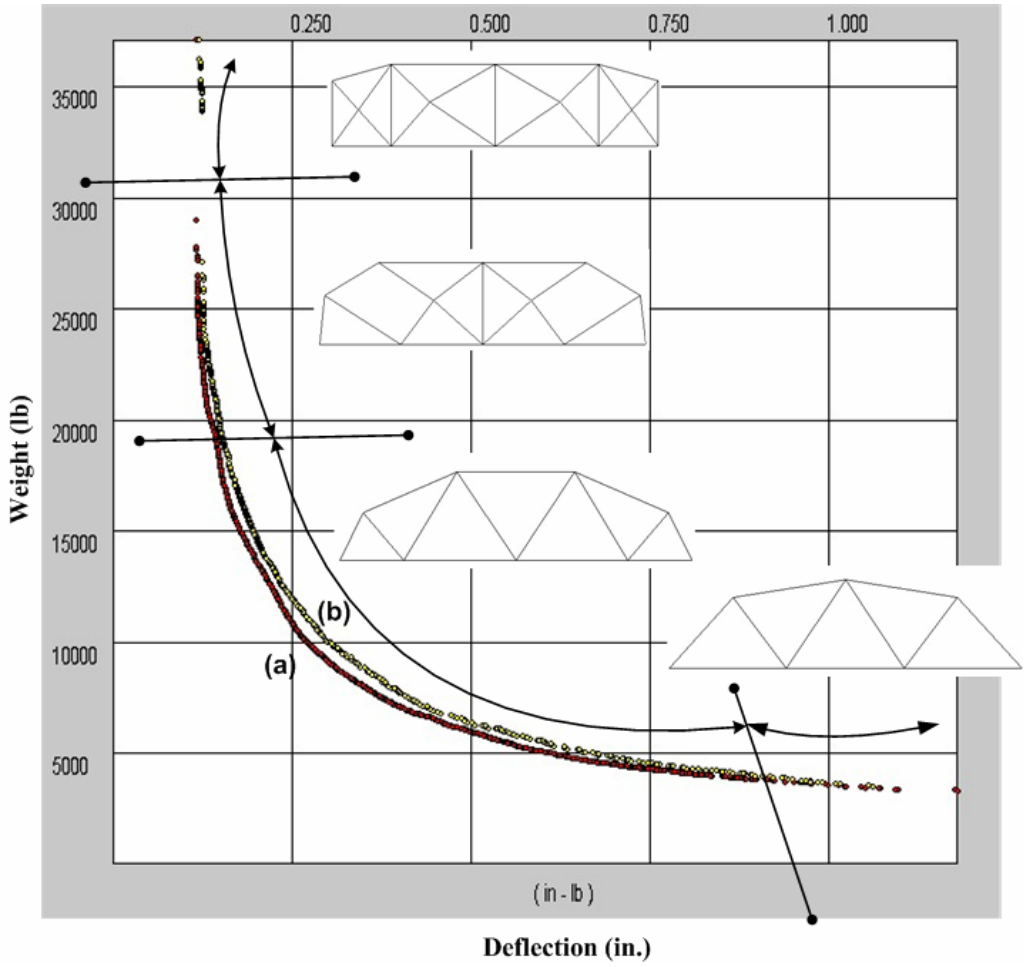
without the restart and sub-process for the 60 ft. span length. For this comparison, the deflection was determined based on the maximum nodal deflection. The results show the improved result by the MOGA algorithm with restart and sub process. Another advantage of the proposed algorithm is that the sensitivity of the parameter settings on the quality of the results is decreased. Using the previous algorithm, the changes of parameters produced different results because the changes caused the initial individuals to explore different regions of the search space.

The early, locally optimized individuals interrupted the improvement of the overall population toward the global optimal solutions.

To verify whether this problem is overcome using the proposed algorithm with sub-process and restart, sizing optimization was performed using the trusses in the save pool of main process. As previously stated, all the representatives of each topology are saved in the save pool. If the proposed algorithm still is having problems with fully optimizing the candidate Pareto solutions, the sizing optimization using the trusses in the save pool may produce better trusses that have different topologies identified on the Pareto front.

In the save pool for the 60 ft. span length there were 65 representatives trusses saved. The nodal locations of the trusses were modified to increase the possibility to obtain better trusses. Consequently, 195 trusses were generated including the original trusses (representatives). After performing sizing optimization on the 195 trusses, the 195 Pareto fronts were obtained, and by identifying the dominant individuals by assigning the rank to each individual of the 195 Pareto fronts, one Pareto front was

obtained. Fig. 5.21 presents the comparison between the results from the proposed algorithms and those obtained from sizing optimization of the modified trusses and the trusses in save pool.



(a) The Pareto front generated from the proposed algorithm.
(b) The Pareto front generated from the sizing optimization using the representatives in save pool.

Fig. 5.21. Verification of the topology of optimized trusses

The Pareto front generated by sizing optimization contained the same topologies as those obtained by the proposed algorithm. In addition, sizing optimization did not provide better results. The same investigation was performed with 40 ft. and 80 ft. span trusses, and the results were similar. Based on these results, the proposed algorithm is able to overcome the problem of having premature individuals removed from the population before they have a chance to be optimized.

The optimization algorithm proposed in this research was not constrained by the biases that a designer picks up in engineering practice. As seen in the results for the three spans, the optimized trusses in the low weight region are in agreement with trusses used in engineering practice. The trusses in the low deflection region, in comparison, have impractical topologies and geometries that may cause a risk in real construction due to the formation of a mechanism under different loading. In reality, the zero force members are necessary for the stability during construction and for the preparation of changes of loads. The identification of even the impractical trusses, however, indicates the flexibility of the representation and algorithm proposed in this research. In Fig. 5.22, the Pareto front composed of the impractical trusses is compared with the Pareto front composed of the modified trusses. To obtain the result, the topology of the truss generated from the proposed algorithm was modified to meet the standard engineering practice. Members were added to convert the quadrilateral elements into triangle elements. In addition, the nodes in which the four members were connected were removed and x-braces were added. Other configurations remained the same as those of the original truss. Sizing optimization was performed with the modified truss. As shown

in Fig. 5.22, the trusses generated from the proposed algorithm show a lower weight and deflection. In the algorithm that was not constrained by standard engineering practice, the MOGA removed the unnecessary members to evolve the better trusses. The removal enabled trusses that had lower weight and deflection to be obtained.

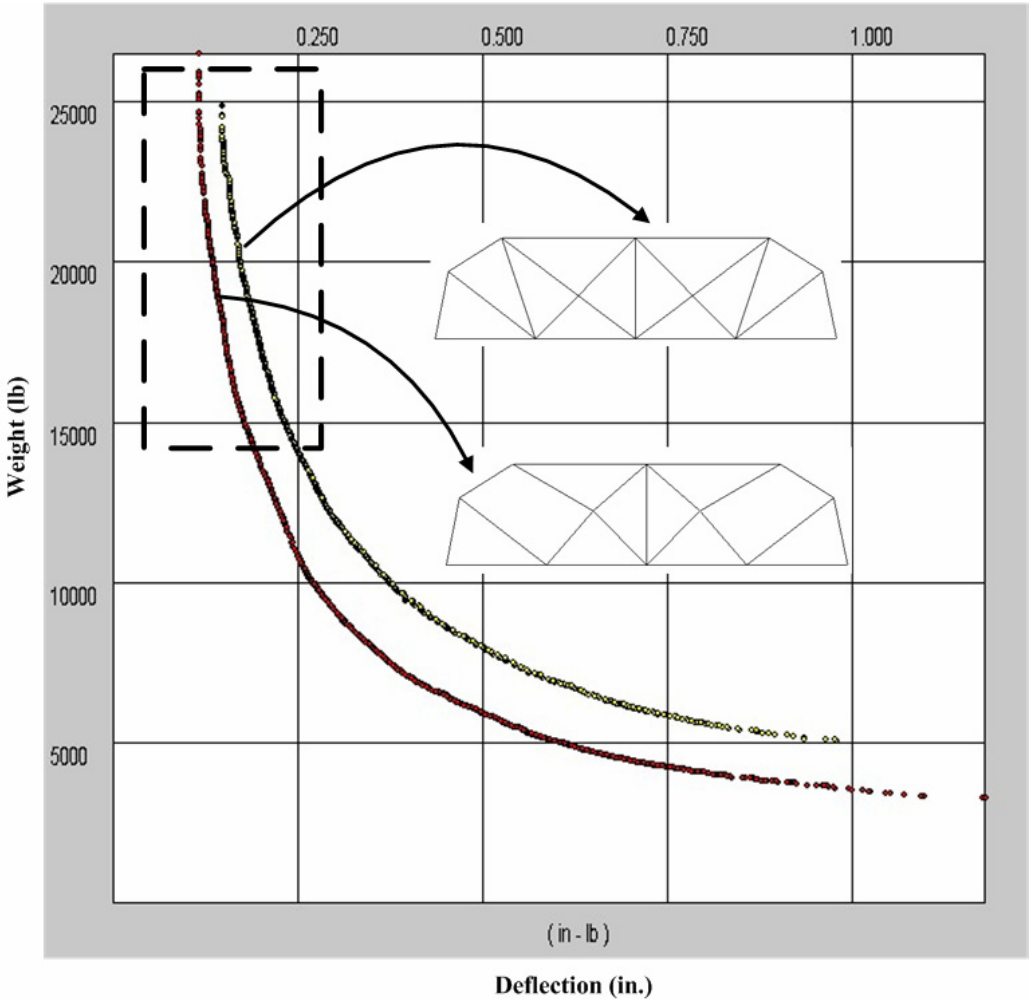


Fig. 5.22. Comparison of topology generated by genetic program with modified topology based on engineering practice

The proposed algorithm fulfilled the flexibility provided by the representation and design grammar proposed in this research and was efficient enough to synthesize design alternatives to find near-optimal trusses.

5.5 Optimizing for Weight, Deflection, and Stress

The proposed algorithm defined by the MOGA with restart and sub-process was applied to the three-objective optimization. A stress objective was added to the previous two objectives. The average stress for stresses of members was computed, and the inverse value of the average stress was minimized. In Fig. 5.23, the three-dimensional Pareto-surface obtained is presented in three different perspectives views.

The increase of the number in conflicting individuals obtained due to the increased number of objectives enables more truss topologies and geometries to be defined on the Pareto surface. The Pareto surface generated was composed of many long and short strips. Each strip was composed of trusses with the same topology and geometry but with changes in member sizes. In Fig. 5.24, the topologies in each strip for each truss defined are roughly indicated.

The increased number of individuals on the Pareto surface caused a shortage of memory allocation and a drop in computational efficiency. The number of non-dominated individuals in Fig. 5.24 is over 10,000. The shortage of memory allocation results in individuals concentrated in the low weight region. Because the individuals in the main pool are sorted by the fitness without consideration of the weight value, the trusses that have lower deflection (high weight) were rarely saved if the pools are full.

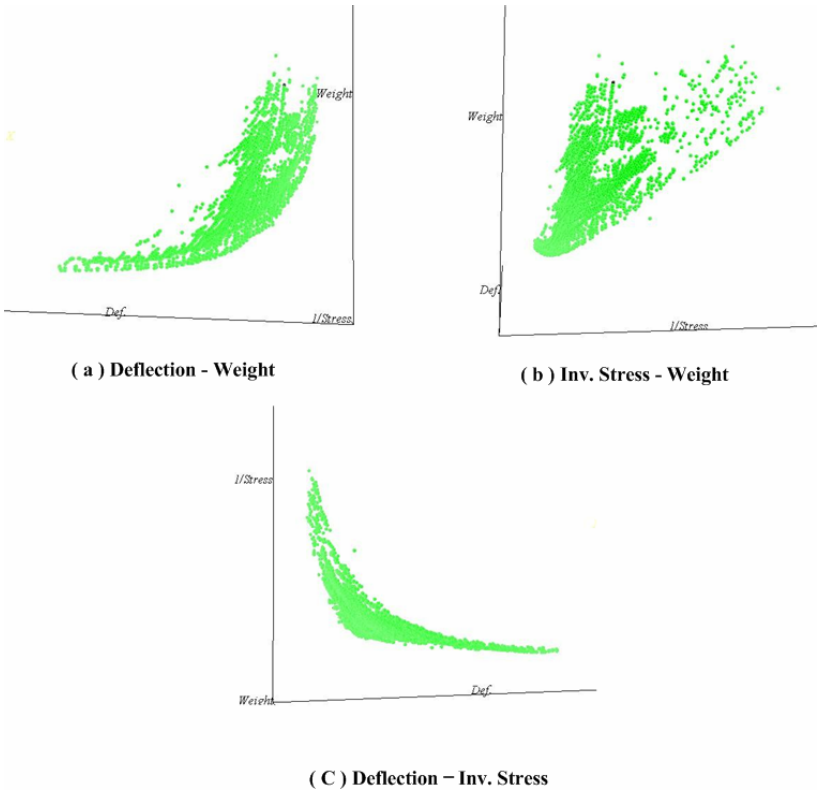


Fig. 5.23. Pareto surface obtained for the 40 ft. span

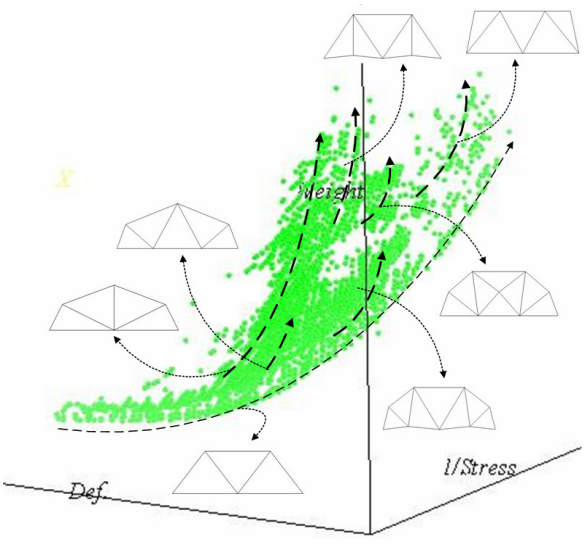


Fig. 5.24. Truss topologies obtained according to regions of the search space for 40 ft. span

5.6 Conclusions

In this Section, a strategy and algorithm for the efficient multi-objective optimization of trusses in design problem domain was developed through several experiments. The final algorithm showed enough efficiency to perform the optimization in the complicated unstructured design domain. The algorithm used a sub-process and restart strategy. The use of the save pool in the main process enabled the premature individuals to have a chance to be optimized without being removed. Even though some of trusses generated in this research did not comply with the standard engineering practice, the algorithm fulfilled the flexibility that is required using the proposed representation and design grammar and was effective in synthesizing designs.

6 COMPARISON OF THE QUALITY OF THE RESULTS WITH LOCAL OPTIMIZATION METHOD, TRIAL, AND ERROR DESIGNS

6.1 Introduction

In Section 5, the modified algorithm using IRR GA was developed and the trial results obtained using the method were presented. In this Section, these results are compared with other optimized trusses by sizing optimization of the trusses identified in Section 5 and with traditional truss configurations used in engineering practice, like the Warren truss. The methodology developed in previous research is introduced and adopted to perform the comparison in this Section.

6.2 Methodology of Comparison

6.2.1 Performance Measure

In a multi-objective optimization problem, measuring the performance of the algorithm is more complex than in single objective optimization problem. In multi-objective problem, the factors that determine the performance can be summarized as follows (Zitzler, Deb, and Thiele 2000):

1. *The distance of the result set to the Pareto-optimal set should be minimized.*
2. *The well distribution of the result set is desired.*
3. *For each objective, a wide range of values should be covered.*

In this research, the results obtained by the proposed algorithm using IRR GA (Raich

1999) were compared with other locally optimized trusses based on the distance measure between non-dominated sets. The measure was assessed using the C function proposed by Zitzler and Thiele (1999).

The two non-dominated sets are compared with each other by calculating the fraction of individuals in each set using the C function. Each comparison is expressed into one scalar value that indicates the performance. The definition of the function is (Zitzler and Thiele 1999):

$$C(X', X'') := \frac{\left|\left\{a'' \in X'', \exists a' \in X' : a' \preceq a''\right\}\right|}{\left|X''\right|} \tag{6.1}$$

'$C(X', X'')$=1' indicates that the all solutions of X'' are dominated by those of X'. By using the C function, the same individuals, which are these shared in both sets, are considered as non-dominated individuals to each other. The comparison in this research was performed based on the C-function. To provide a more detailed illustration of the comparison, however, the individuals that were shared by both sets were expressed into the other independent measure. Fraction of Non-dominated individuals is:

$$(X', X'') := \frac{\left|\left\{a'' \in X'', \exists a' \in X' : a' \prec a''\right\}\right|}{\left|X''\right|} \tag{6.2}$$

Fraction of shared individuals is:

$$(X', X'') := \frac{\left|\left\{a'' \in X'', \exists a' \in X' : a' = a''\right\}\right|}{\left|X''\right|} \tag{6.3}$$

6.2.2 Procedure of Comparison

The results obtained by the proposed algorithm using the IRR GA (Raich 1999) were compared with other result sets to verify the quality of the optimization. Through the comparisons, the performance of proposed algorithm was evaluated. The objective of this research is to perform the sizing, shape, and topology optimization at the same time in the unstructured design domain. Therefore, the quality of the results obtained by the proposed algorithm was verified by the following procedures:

1. Check the degree of sizing optimization

The trusses on the non-dominated front obtained by the proposed algorithm are locally optimized (sizing optimization). The Pareto sets obtained by sizing optimization are compared with the non-dominated front.

2. Check the degree of geometry optimization

The nodal locations of the trusses that are on the non-dominated front obtained by the proposed algorithm are arbitrarily modified, and the modified trusses are locally optimized (sizing optimization). The sets generated by sizing optimization are compared with the non-dominated front.

3. Check the degree of topology optimization

(1) The trusses in the save pool are modified and locally optimized (sizing optimization). The sets generated by the sizing optimization are compared with the non-dominated front

obtained by the proposed algorithm. This comparison was already presented in Section 5 to investigate whether the premature individuals were prevented from being removed during optimization.

(2) The trusses that are designed based on the engineering practice are locally optimized (sizing optimization). The Pareto sets obtained by sizing optimization are compared with the non-dominated front obtained by the proposed algorithm.

6.3 Investigate Performance of the Proposed Algorithm by Comparison

6.3.1 Local Refinement

To investigate the degree of local optimization (sizing optimization) performed, the trusses on the non-dominated front obtained by the proposed algorithm were subjected to sizing optimization. Each truss that had a different geometry and topology on the non-dominated front was locally optimized. The obtained Pareto fronts for each individual truss topology and shape were superimposed to identify the non-dominated individuals for all the shapes and topologies. Finally, one non-dominated front was obtained.

Fig. 6.1 presents the non-dominated front obtained from the proposed algorithm and from the local refinement of the results for the 60 ft. span truss. As shown in Fig. 6.1, the two fronts completely coincide. In addition to the 60 ft. span, the comparisons for the 40 ft and 80 ft. spans also show that the two fronts exactly coincide. This

comparison showed the results obtained by the proposed algorithm were locally well optimized.

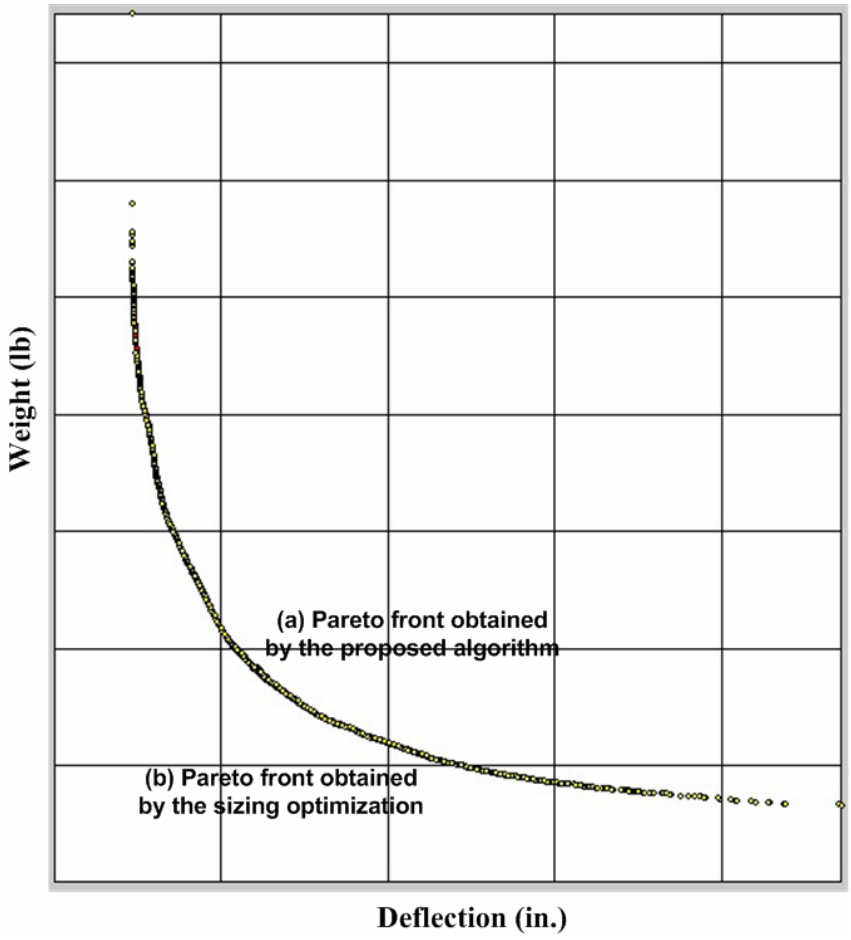


Fig. 6.1. Non-dominated front from the proposed algorithm and the local refinement (60 ft. span)

6.3.2 Geometry Optimization

From current research, it has been stated that the change of nodal locations caused variation in the efficiency of truss design. This means that the changes in nodal

location may cause an enhancement in the optimality of a truss structure.

To evaluate the degree of shape optimization performed by the proposed method, each different topology and shape on the non-dominated front obtained by the proposed algorithm was selected. The nodal locations of the selected trusses were arbitrarily modified. In this manner, four sets that consist of several topologies were generated for each span size. Fig. 6.2 presents the four truss sets for 40 ft. span. The trusses in each set were subject to sizing optimization. Consequently, four non-dominated fronts were obtained. The results obtained by the proposed algorithm were compared with each of the four non-dominated fronts.

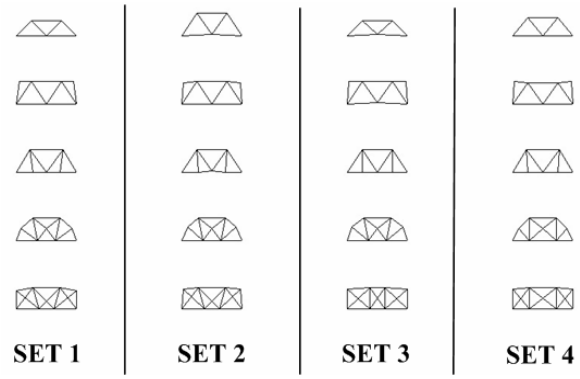


Fig. 6.2. Modified trusses generated based on the trusses on the Pareto front obtained by the proposed algorithm (40 ft.)

As illustrated in Fig. 6.3, except for the region where both objectives are minimized, the two curves almost coincide. To further investigate the performance of the proposed algorithm, the comparison was also performed by using the modified C function.

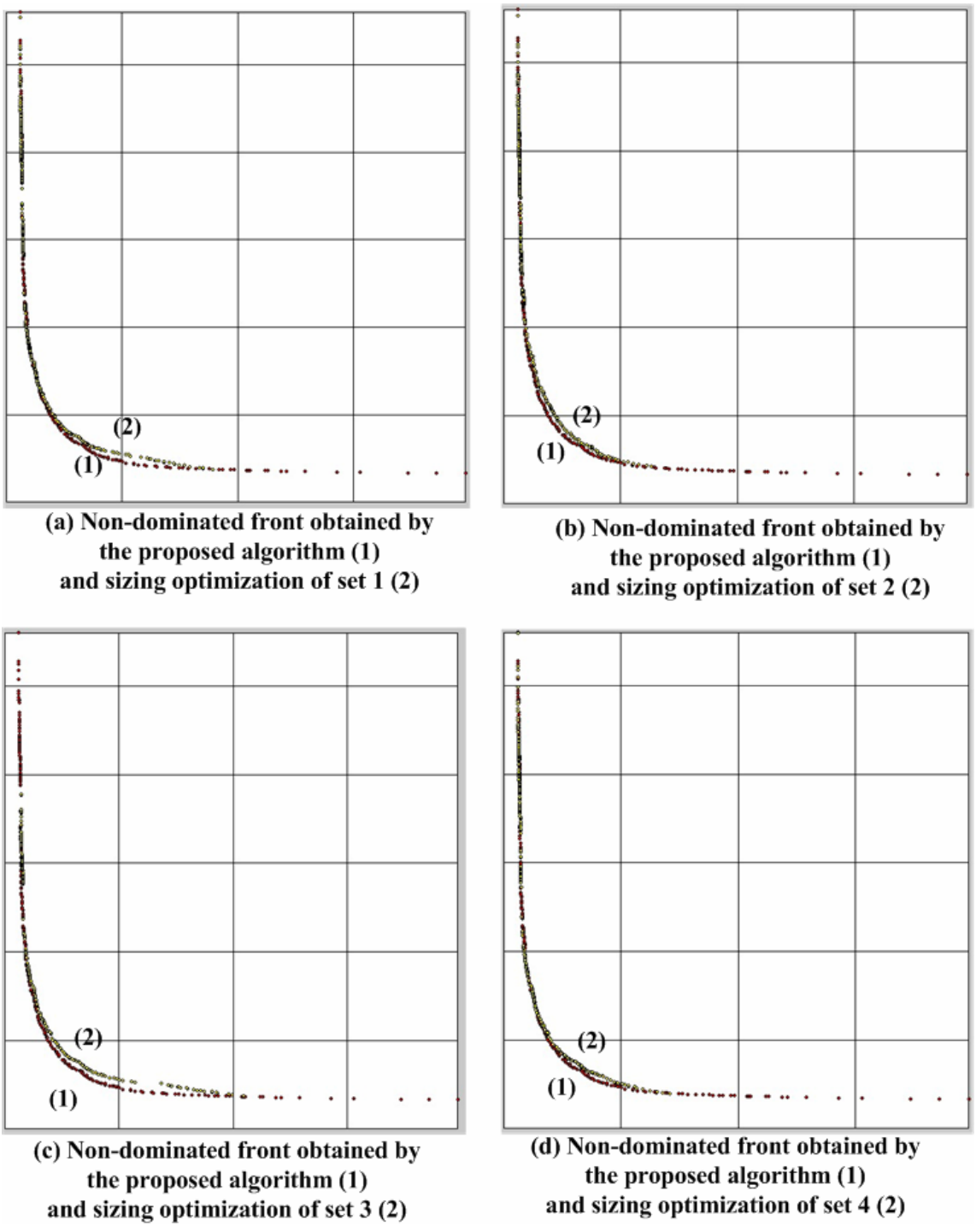
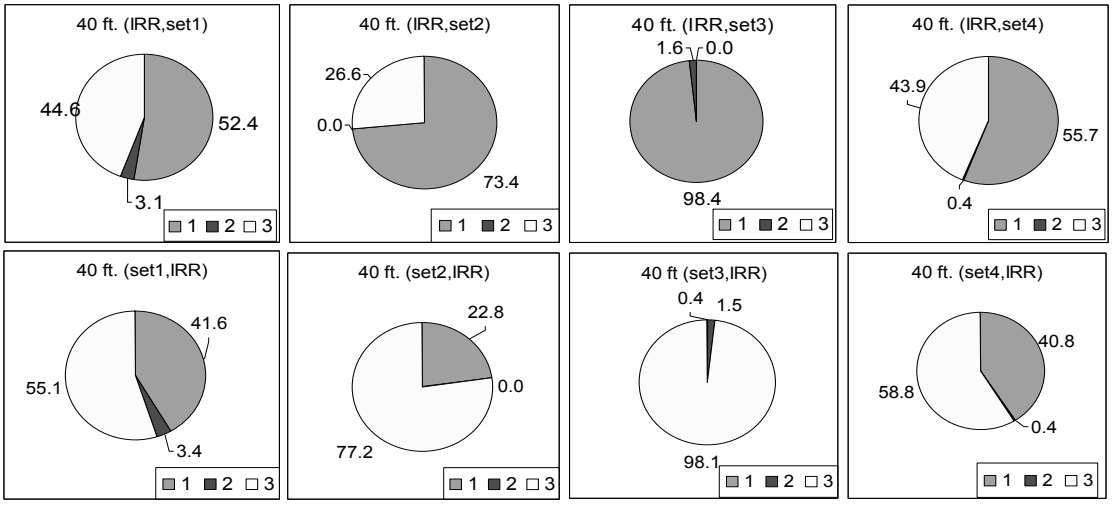


Fig. 6.3. Non-dominated fronts in 40 ft. span obtained by the proposed algorithm and the sizing optimization of four set

Fig. 6.4 presents the percentage of non-dominated, dominated, and shared individuals between two curves. In all the comparisons, the fraction of individuals shared by two curves is under 4 percent. Individuals obtained by the proposed algorithm dominated those obtained from the sizing optimization by at least 50 percent. For the case of comparison with set 1, the error corresponding to each deflection and weight range is presented in Fig. 6.5. The deflection and weight errors between two individuals, which were on different curves, were computed. The error for one objective was computed while the other objective of two individuals was held. However, individuals, that have the same deflection and/or weight as each other on the two different curves, are rarely found. Therefore, if the difference of objective values between individuals on two different curves is within 0.5 percent, the values were assumed to be the same.

In Fig. 6.5, positive values mean that the individuals generated by the proposed algorithm dominate those generated by the sizing optimization. Reversely, negative values mean the individuals obtained by the sizing optimization dominated those obtained by the individuals obtained by the proposed algorithm. The worst case occurred in the comparison with set 1. The results obtained by the proposed algorithm dominated the locally optimized set by only 52.4 percent. The weight error based on the same deflection showed that the results from the proposed algorithm were mostly superior to the trusses in set 1, and the deflection error based on the same weight showed the results from the proposed algorithm mostly are dominated by the trusses in set 1. In any case, the negative errors in latter case were within only 6 percent, which is not a considerable error relative to the magnitude of the positive values.



$$(X', X'') := 1: \frac{|\{a'' \in X'', \exists a' \in X' : a' \prec a''\}|}{|X''|} \quad 2: \frac{|\{a'' \in X'', \exists a' \in X' : a' = a''\}|}{|X''|}$$

$$3: \frac{|\{a'' \in X'', \exists a' \in X' : a' \succ a''\}|}{|X''|}$$

Fig. 6.4. Comparison of the results (40 ft. span) with locally optimized trusses

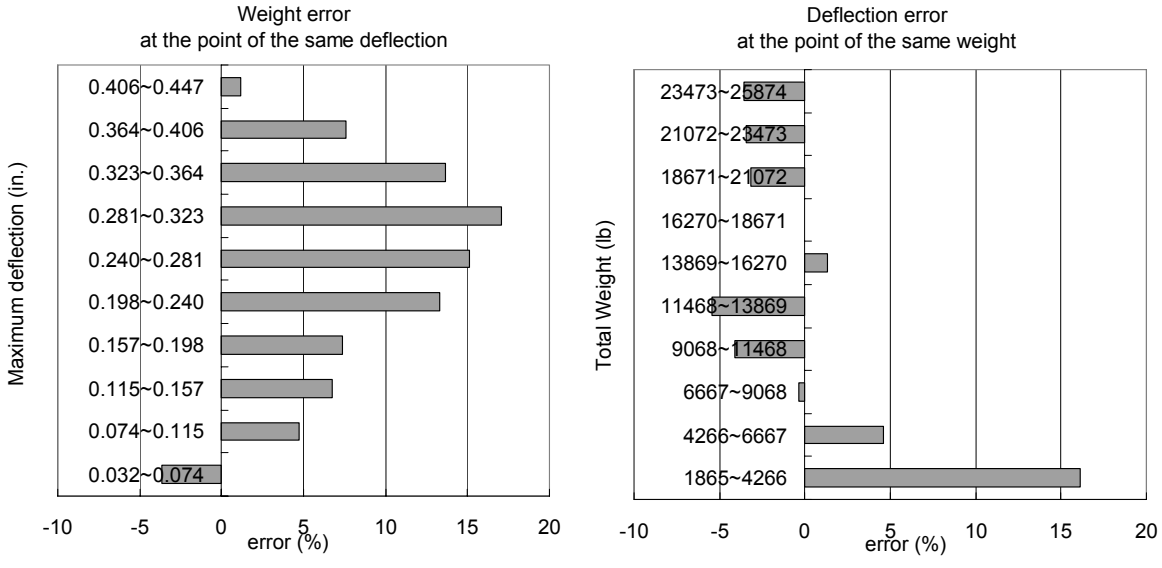
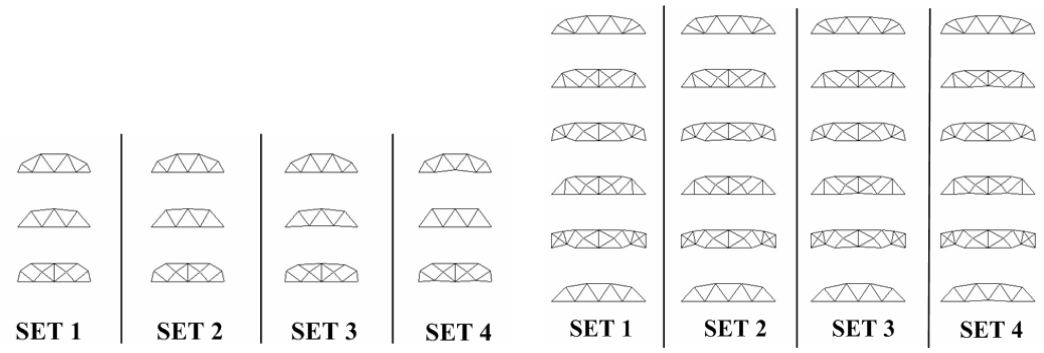


Fig. 6.5. Weight and deflection errors based on the sizing optimization of set 1

The results for 60 ft. and 80 ft. spans were subject to the same procedures for comparison as these used for the 40 ft. span. Fig. 6.6 presents the four truss sets for 60 ft. and 80 ft. span. As results, Fig. 6.7 to 6.12 present the comparisons with the locally optimized truss results. Fig. 6.8 shows that the result obtained by the proposed algorithm is primarily dominant except for the comparison with set 1.

In the comparison with set 1, only 27.5 percent of the results dominated the locally optimized trusses (Fig. 6.8). However, the negative errors illustrated in the Fig. 6.9 did not exceed 3 percent. The comparison with other sets showed the results in this research were completely dominant. In the comparisons performed for the 80 ft. span, the two sets were shown to be very close to each other. In the range between 67.7 and 94.4 percent, the results dominate the locally optimized trusses and the negative errors were approximately one percent.



(a) 60 ft. span (b) 80 ft. span

Fig. 6.6. Modified trusses generated based on the trusses on the Pareto front obtained by the proposed algorithm

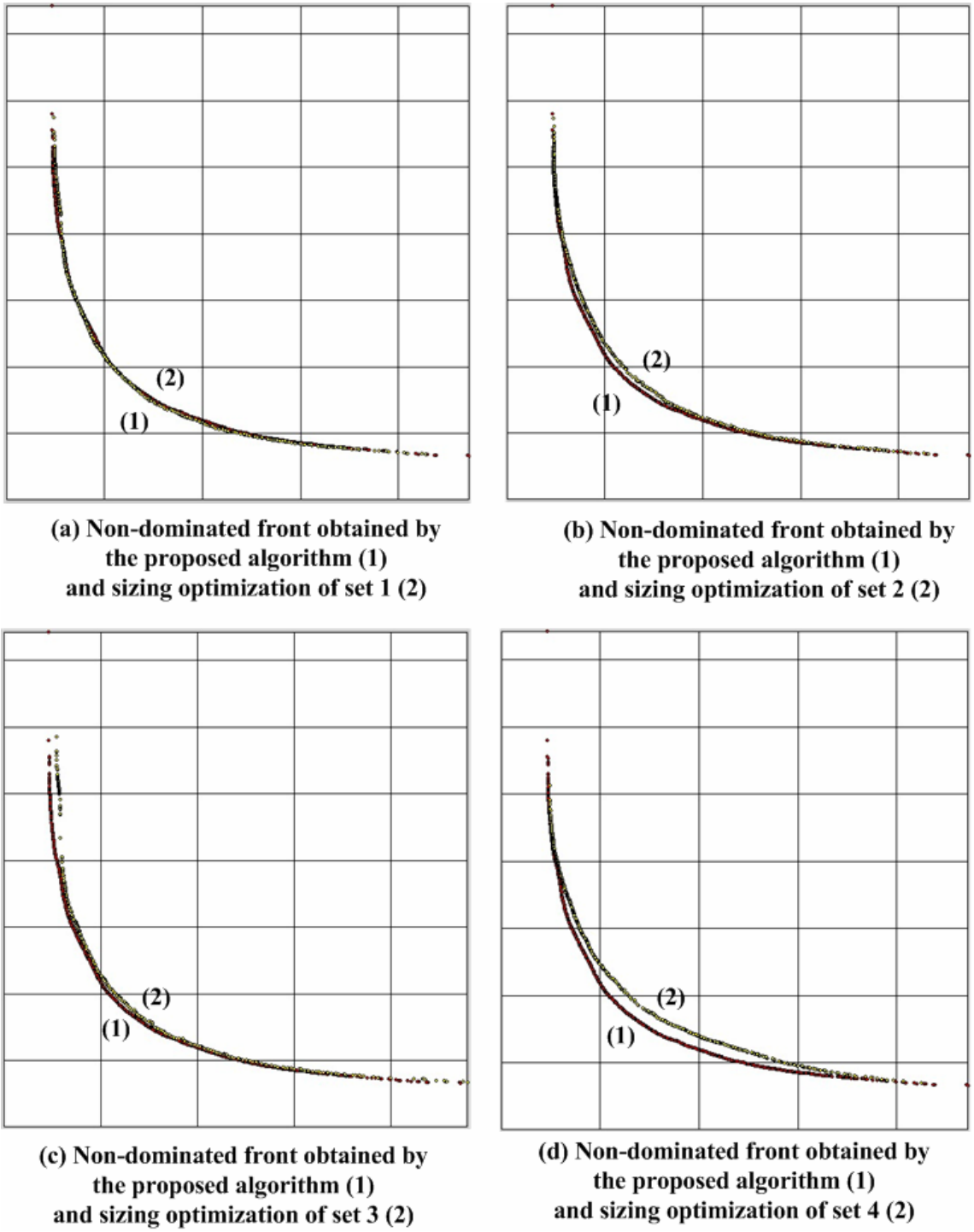
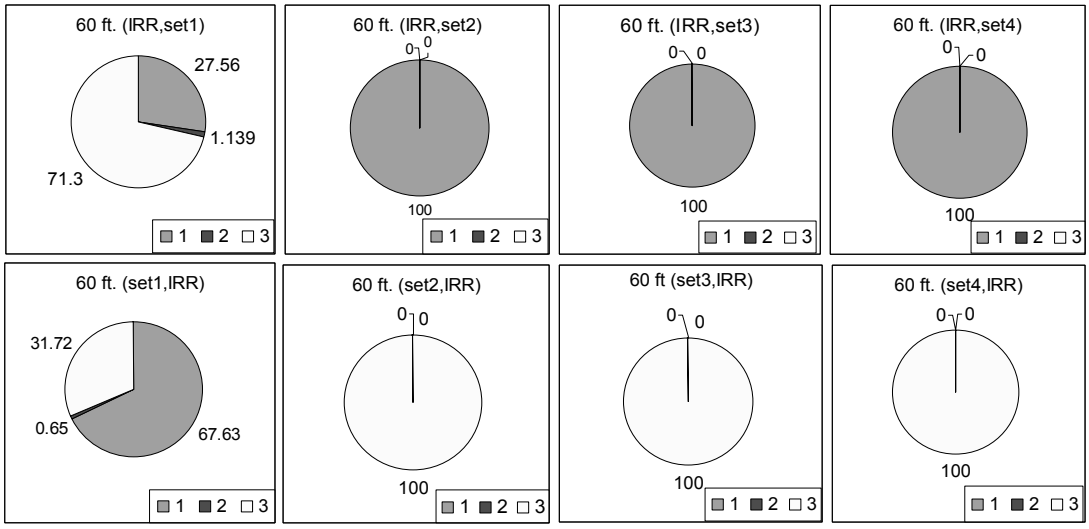


Fig. 6.7. Non-dominated fronts in 60 ft. span obtained by the proposed algorithm and the sizing optimization of four sets



$$(X', X'') := 1: \ \frac{\left|\left\{a'' \in X'', \exists a' \in X' : a' \prec a''\right\}\right|}{\left|X''\right|} \quad 2: \ \frac{\left|\left\{a'' \in X'', \exists a' \in X' : a' = a''\right\}\right|}{\left|X''\right|}$$

$$3: \ \frac{\left|\left\{a'' \in X'', \exists a' \in X' : a' \succ a''\right\}\right|}{\left|X''\right|}$$

Fig. 6.8. Comparison of the results (60 ft. span) by calculating the fraction of individuals

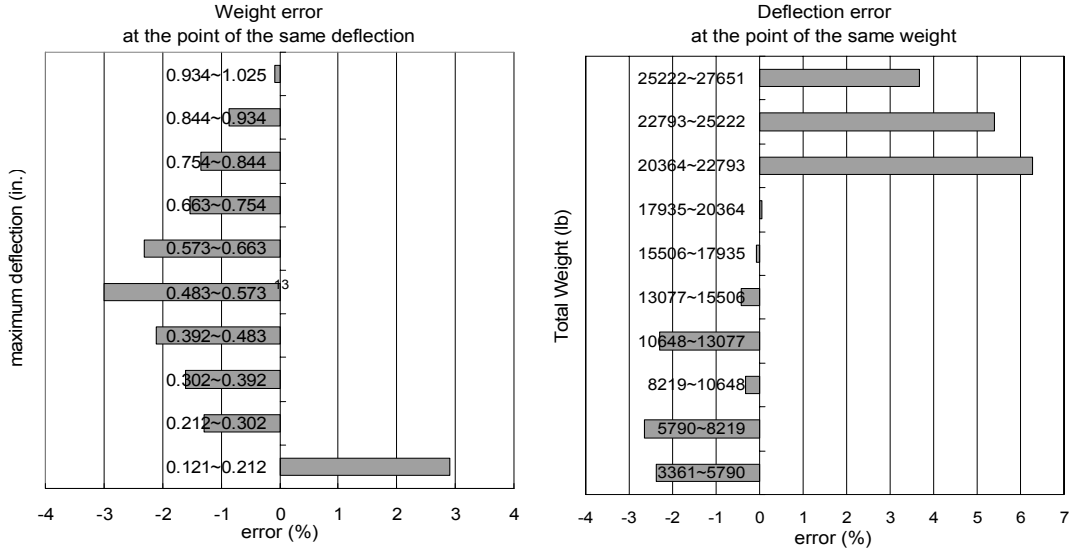


Fig. 6.9. Weight and deflection errors based on the sizing optimization of set 1

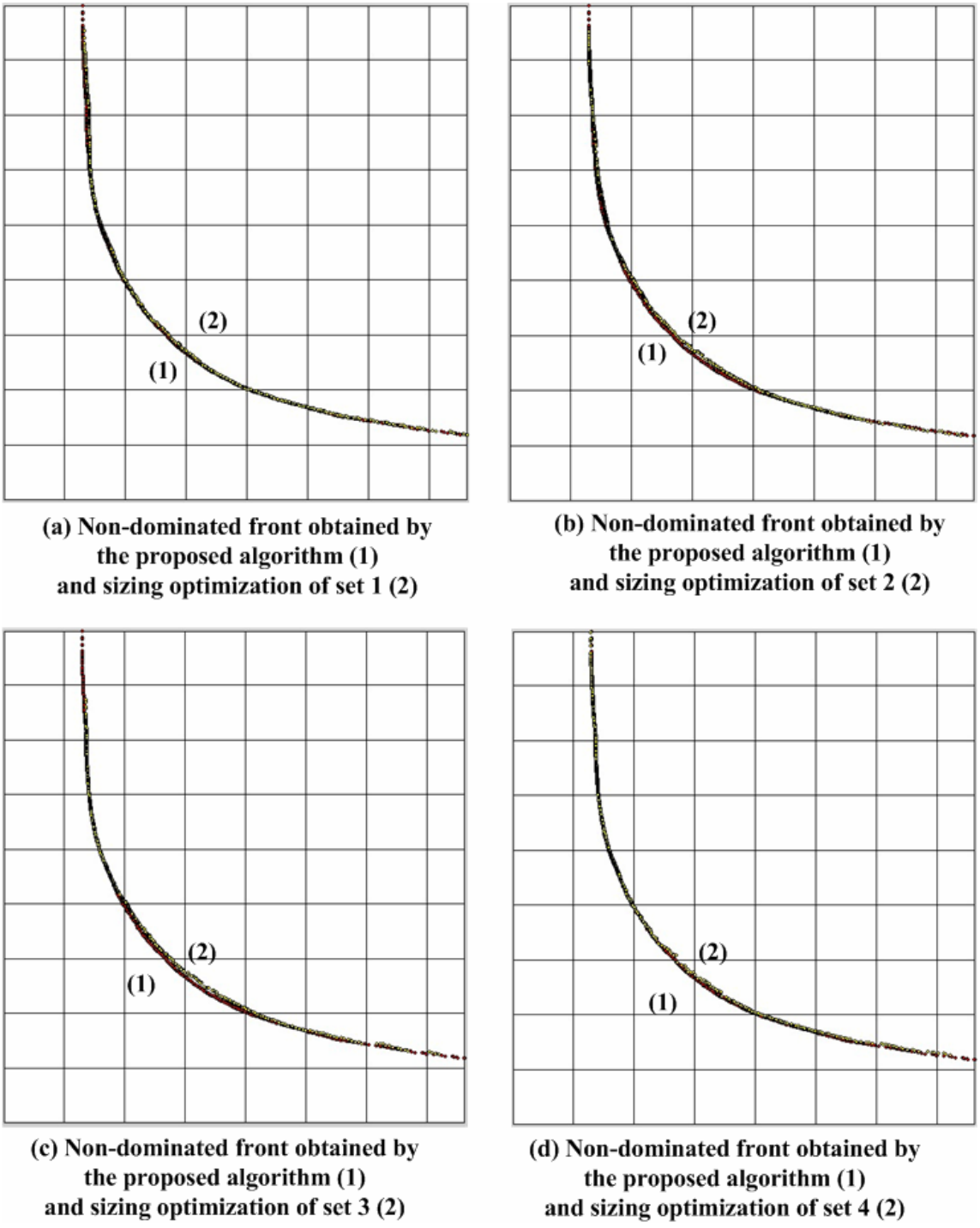
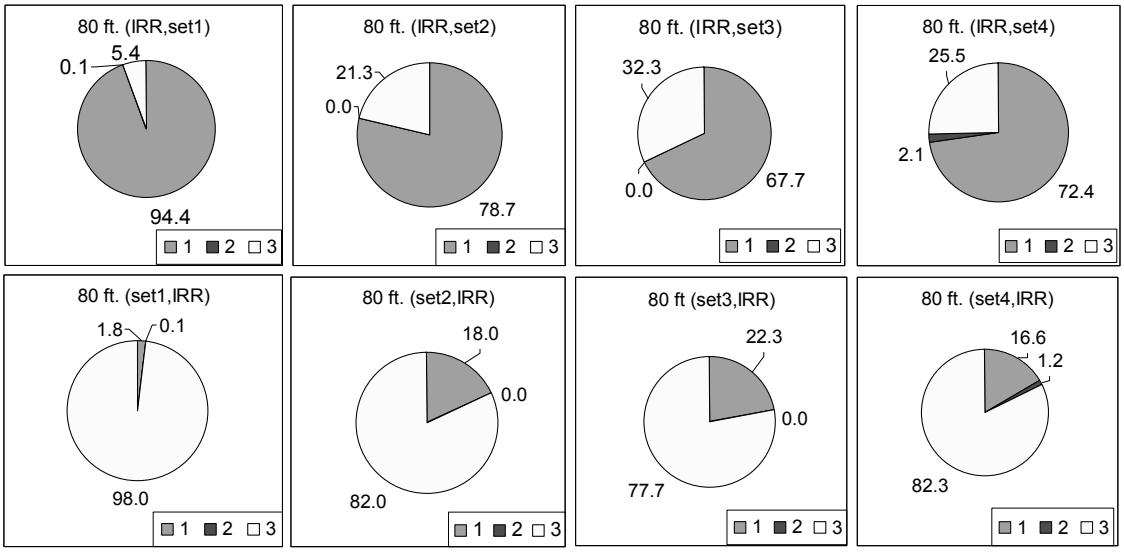


Fig. 6.10. Non-dominated fronts in 80 ft. span obtained by the proposed algorithm and the sizing optimization of four sets



$$(X', X'') := 1: \frac{\left|\{a'' \in X'', \exists a' \in X' : a' \prec a''\}\right|}{|X''|} \quad 2: \frac{\left|\{a'' \in X'', \exists a' \in X' : a' = a''\}\right|}{|X''|}$$

$$3: \frac{\left|\{a'' \in X'', \exists a' \in X' : a' \succ a''\}\right|}{|X''|}$$

Fig. 6.11. Comparison of the results (80 ft. span) by calculating the fraction of individuals

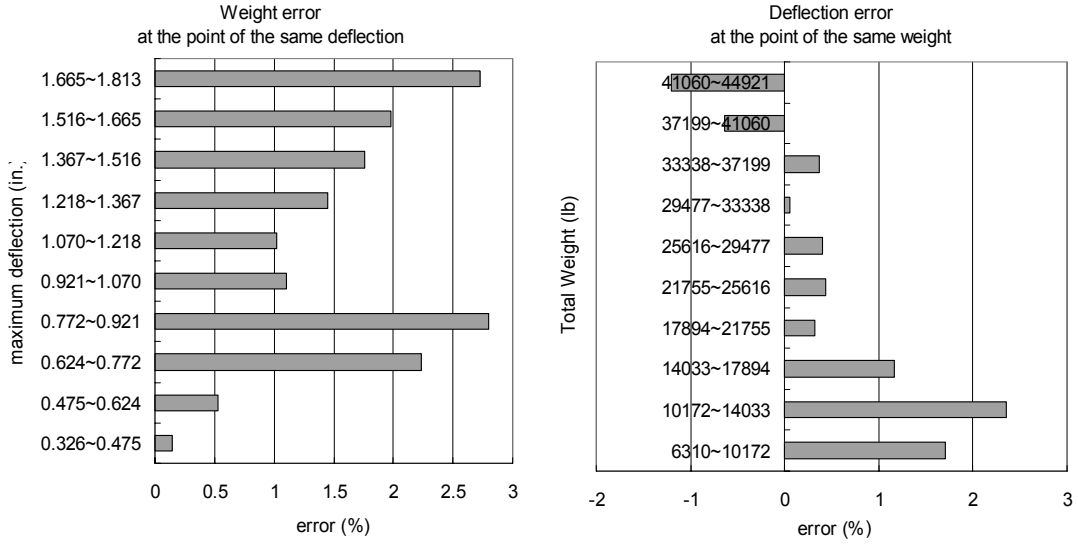


Fig. 6.12. Weight and deflection errors based on the sizing optimization of set 4

The results obtained by the proposed algorithm using IRR GA were consistently better than the results obtained for the locally optimized trusses. As stated at the beginning of this comparison, the seeds of the sizing optimization were composed of the modified trusses based on those that were already optimized by the proposed algorithm. Considering this fact, the performance of the proposed algorithm using IRR GA was considered to be effective and was able to maintain the flexibility required during the shape evolution of a truss.

6.3.3 Topology Optimization

In Section 5, sizing optimization using the seeds, which were modified trusses based on the trusses in the save pool, was performed to verify that the algorithm preserved the premature individuals and facilitated the optimization of the premature individuals. As the result of the comparison, none of the new topologies considered were found on the non-dominated front. This means that the proposed algorithm was effective in performing topology optimization. In this Section, the degree of topology optimization of the results is investigated again using trusses that are designed based on engineering practice.

For the investigation, several types of trusses, such as Pratt, Howe, and Warren trusses, were designed. The nodal locations of each truss were arbitrarily modified. Fig. 6.13 and Fig. 6.14 present the manually designed truss topologies along with the modified truss topologies. Like the investigation undertaken to evaluate the degree of shape optimization, the trusses were subject to sizing optimization. Each truss topology

and geometry formed a different Pareto front. Fig. 6.15 shows the different Pareto fronts obtained by performing sizing optimization of each topology and shape.

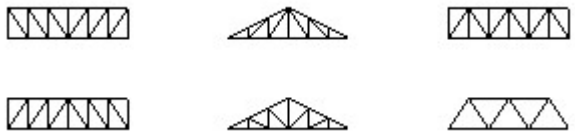


Fig. 6.13. Truss designs developed based on engineering practice (60 ft. span)

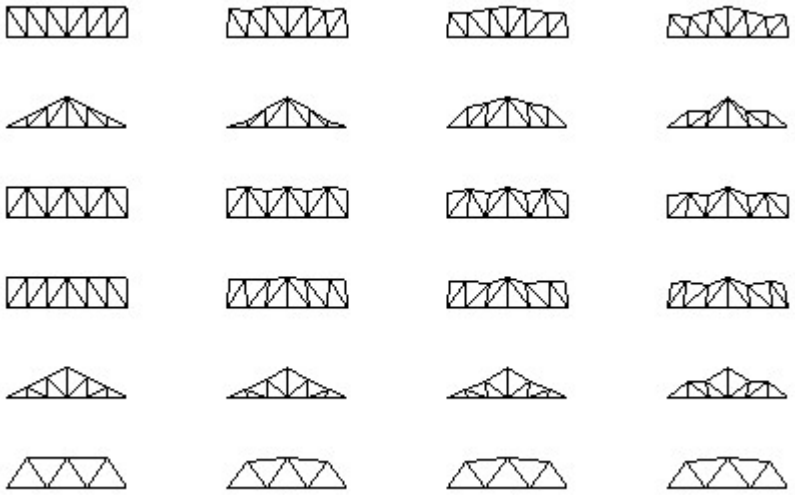


Fig. 6.14. Modified trusses based on the truss topologies in the Fig. 6.13

According to changes in member sizes, each topology forms a separate Pareto front. Among the designed topologies, the Warren truss is shown to be superior to the other truss topologies. Fig. 6.16, in addition, illustrates that small changes in nodal locations create new Pareto fronts. The alternation of any truss designs through changing nodal

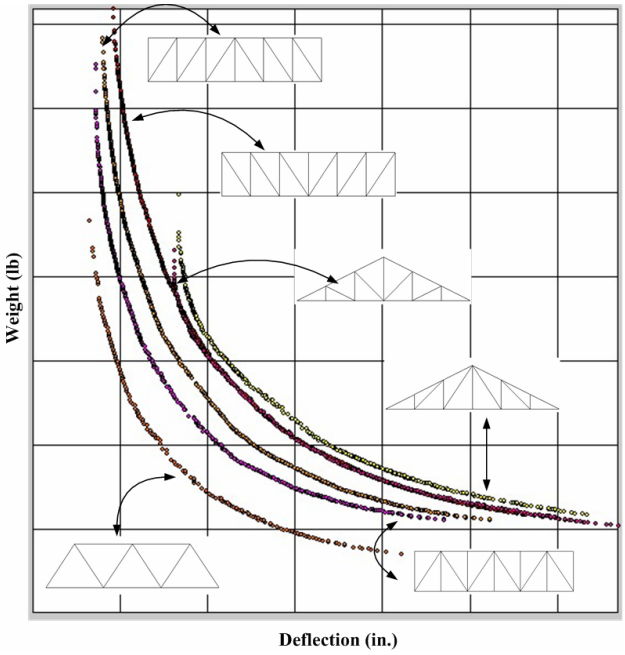


Fig. 6.15. Non-dominated fronts for each topology (60 ft. span)

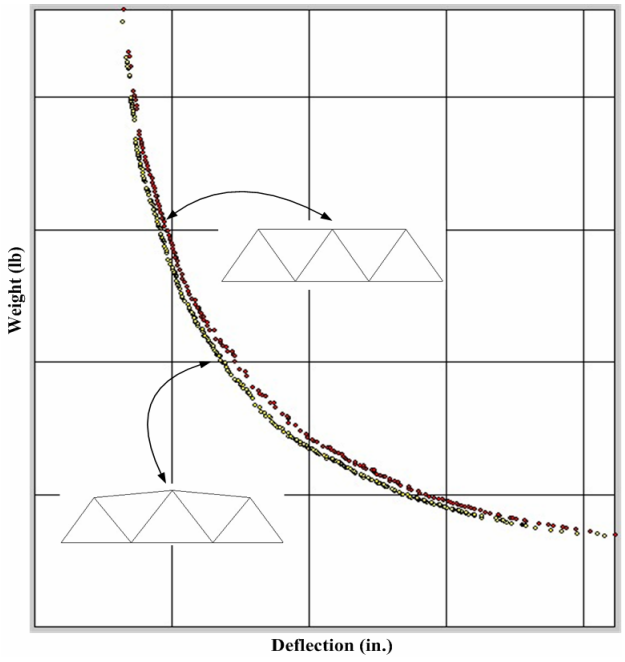


Fig. 6.16. Non-dominated fronts of the same topology with different nodal locations (60 ft. span)

locations and topology creates another Pareto front. This means that each alternation creates local optima. Considering changes in topology, shape, and member sizes, the search space for this problem domain is extremely complicated and large.

The Pareto front generated for the Warren truss configuration dominated all of the other Pareto fronts generated for the other trusses designed based on engineering practice. The final non-dominated front, which was composed of Warren trusses, was compared with the results obtained by the proposed algorithm. In the region of low weight, the two fronts are shown to be similar. In that region, the topologies of both curves are Warren type trusses. Fig. 6.17 presents the comparison of two curves.

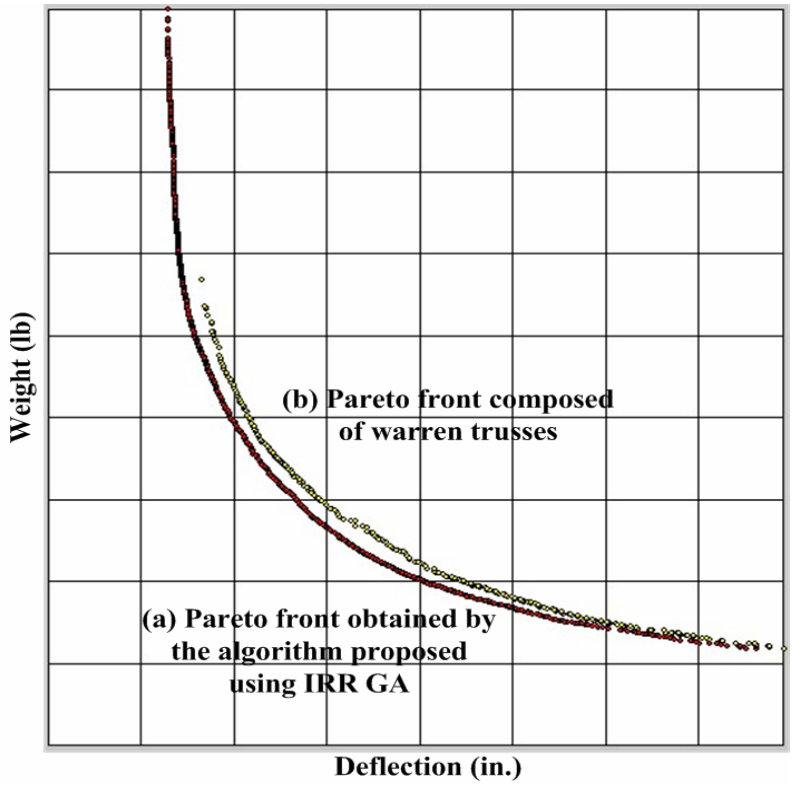
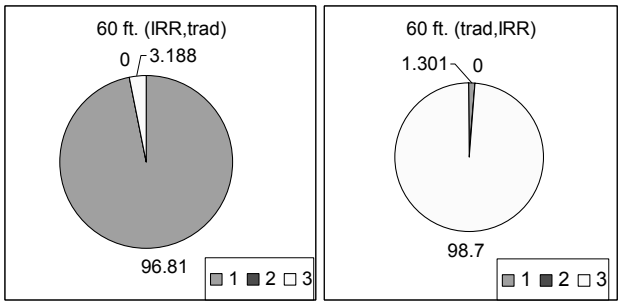


Fig. 6.17. Comparison-1 with trusses designed based on engineering practice (60 ft. span)

The result obtained by the proposed algorithm dominated the trusses designed based on engineering practice by 96.8 percent as shown in Fig 6.18.



$$(X', X'') := 1:\ \frac{\left|\{a'' \in X'', \exists a' \in X' : a' \prec a''\}\right|}{|X''|}\quad 2:\ \frac{\left|\{a'' \in X'', \exists a' \in X' : a' = a''\}\right|}{|X''|}$$

$$3:\ \frac{\left|\{a'' \in X'', \exists a' \in X' : a' \succ a''\}\right|}{|X''|}$$

Fig. 6.18. Comparison-2 with trusses designed based on engineering practice (60 ft. span)

The comparison study for the 80 ft. span length was performed in the same manner as that used for 60 ft. span. The comparison results are also similar to those for the 40 ft span trusses. Fig. 6.19 and 6.20 show the truss designs and the trusses modified arbitrarily by changing nodal locations for the 80 ft. span. Each topology formed an individual Pareto curve. Fig. 6.21 presents the Pareto fronts obtained for each truss topology with Warren type trusses dominating the other types of trusses. For all three spans, the Warren trusses were dominant. Even in the Pareto front obtained by the proposed algorithm, Warren trusses were always present in the region of low weight. The efficiency of Warren truss in this comparison, however, may be limited to the

member selection set used.

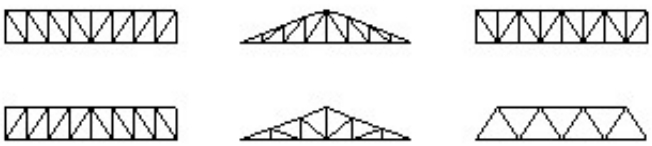


Fig. 6.19. Truss designs based on engineering practice for 80 ft. span

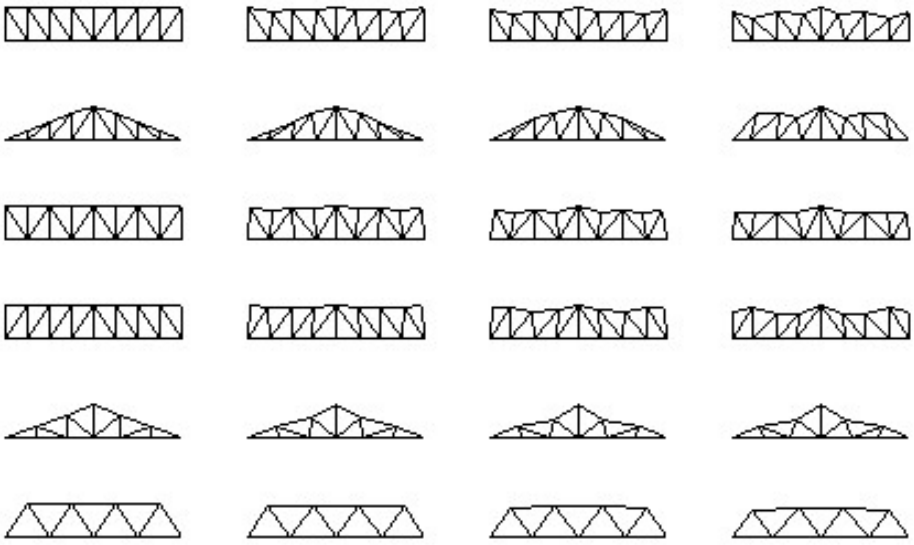


Fig. 6.20. Modified trusses based on the trusses in the Fig. 6.19

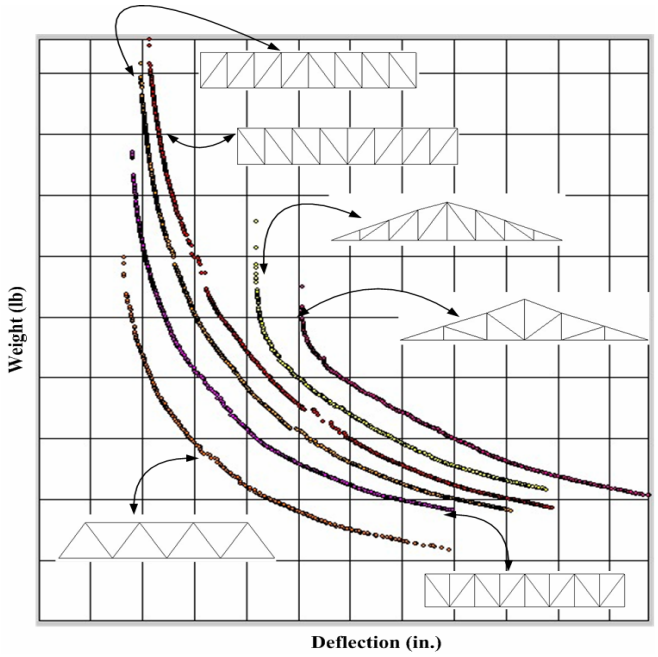


Fig. 6.21. Non-dominated fronts for each topology (80 ft. span)

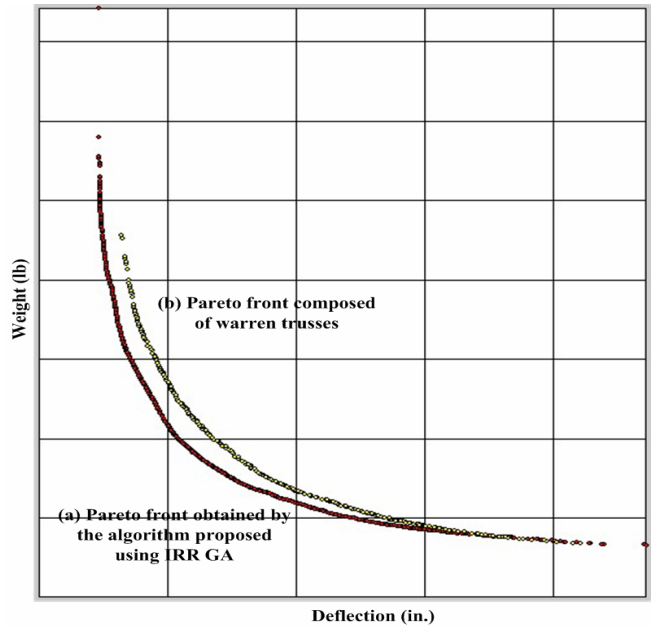
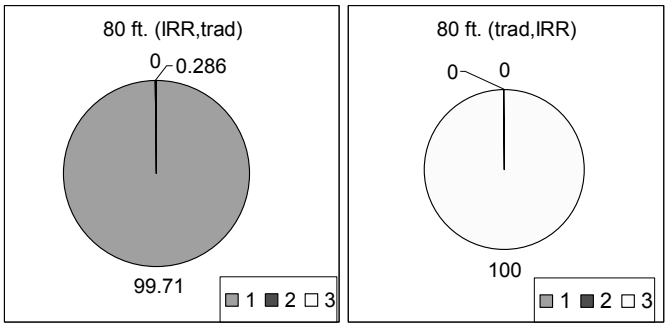


Fig. 6.22. Comparison-1 with trusses designed based on engineering practice (80 ft. span)

In Fig. 6.22, in the region of low weight, in which the Warren trusses were presented, the two fronts merged. The trusses generated by the proposed algorithm using IRR GA dominated the trussed designed based on engineering practice by 99.7 percent with the rest of trusses shared by both curves as shown in Fig 6.23.



$$(X', X'') := 1:\ \frac{\left|\{a'' \in X'', \exists a' \in X' : a' \prec a''\}\right|}{\left|X''\right|}\quad 2:\ \frac{\left|\{a'' \in X'', \exists a' \in X' : a' = a''\}\right|}{\left|X''\right|}$$

$$3:\ \frac{\left|\{a'' \in X'', \exists a' \in X' : a' \succ a''\}\right|}{\left|X''\right|}$$

Fig. 6.23. Comparison-2 with trusses designed based on engineering practice (80 ft. span)

6.4 Conclusions

In this Section, the results obtained by the algorithm proposed using the IRR GA were compared with those generated by sizing optimization to verify the degree of sizing, shape, and topology optimization. Even though it was considered that the algorithm experienced difficulty in finding the best nodal locations in geometry evolution, the comparisons results on the whole showed that the proposed algorithm was

very effective in optimizing the unstructured design domain. The results obtained for the three spans investigated were considered as providing near-optimal Pareto sets within the defined constraints.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The objective of this research was to develop a computational method that can design a set of near-optimal, efficient roof trusses. To meet this objective, several algorithms with various diversity preserving strategies were investigated.

As a first step, an investigation to find optimal GA parameter settings was performed. Unlike other types of GAs, the implicit redundant representation GA (IRR GA) (Raich 1999) used in this research has redundant segments, which positively affect the efficiency and flexibility of algorithm. As discussed in Section 4, trials were performed to find the optimal length of the string that included redundant segments. In addition, the trials discussed in Section 3 helped to understand the characteristic of the unstructured problem domain searched in this research.

Several algorithms were proposed in Section 5. The definition of a large, complex problem domain made it difficult for the individuals in the GA population to explore the search space. In this complicated problem domain, the results usually responded very sensitively to the changes in the GA parameters. The first two proposed algorithms could not preserve the presence of premature individuals in the population for further optimization, which affected the quality of the Pareto front obtained.

The final proposed algorithm with the restart strategy and sub-processes was very effective in searching the complex unstructured problem domain. By using a save pool and a backup pool, the premature individuals could be preserved. The restart

strategy helped maintain the premature individuals and provided opportunities for further optimization.

The algorithm is composed of the main process and sub process. In the main process, the seeds and the results for each sub-process are managed. The main process is composed of three pools:

1. rank pool: save the ranked individuals such as in the external set in SPEA (Zitzler and Thiele 1999)
2. Save Pool: save and cumulate the individuals in the rank pool of sub process. The saved individuals are used as seeds of the sub processes.
3. Backup Pool: save each different topology of trusses.

The genetic process is executed in the sub-process. Unlike other GAs, each sub- process also has the three pools:

1. Mating pool: the pool for GA operation.
2. Rank pool: save the ranked individuals
3. Backup pool: save each different topology of trusses.

In this proposed algorithm, the niche count is computed using a sharing function (Goldberg 1989) and the Strength concept (Zitzler and Thiele 1999). Sharing is formulated in terms of the topology and shape of a truss, and the Strength is computed in terms of objective values. The pools for preserving premature individuals in this complicated search domain and the combination of sharing function and strength for efficient exploration resulted in quality Pareto fronts as results.

In Section 6, the results obtained by the proposed algorithm were compared

with the other locally optimized trusses to investigate the degree of sizing, geometry, and topology optimization performed. The comparison showed that the performance of the proposed algorithm, in which the sizing, shape, and topology optimization were simultaneously performed, was very effective even in the complex unstructured design domain stated.

7.2 Future Work

The main purpose of the representation and algorithm proposed in this research is to maintain the encoding flexibility, so that diverse topology and geometries of trusses can be obtained. However, the excessive flexibility of the representation caused many problems. The flexible representation created a huge search space that had a relatively small feasible area based on the trial results. In addition, the trusses on the Pareto fronts obtained in this research were sometimes far from the trusses designed by engineering practice. Therefore, even though the obtained trusses are well optimized with respect to weight and deflection, they would not be able to be directly used in the real construction without some modification. Another problem is that the computational expense is high. The excessive flexibility generated many topologies and each topology was assigned to one independent sub-process. Consequently, there were a large number of sub-processes that caused the runtime to be slower. In this research, the proposed algorithm was implemented using a MFC-based windows operating system. In order to ensure the practical use of this approach for optimization, the computational run-time must be decreased.

To solve the problems that were identified by this research, the following future research direction are recommended:

1. *Develop an efficient and practical design grammar and representation*

   For efficient searching, the design grammar and representation are important. Looking previous efforts, many design grammars have been introduced. The design grammar, however, that is efficient as well as flexible is required. For instance, a design grammar using triangle elements or predefined geometries will not create unstable trusses, and consequently, the search domain could be substantially reduced. However, the flexibility of the design grammars should be investigated. In order to obtain a near-global optimal set, having a shortage of flexibility would prevent the GA from synthesizing the design alternatives effectively.

2. *Use parallel computing on a UNIX-based system.*

   To reduce the computational time, the use of parallel computing systems is recommended. The algorithm proposed in this research has an architecture that is suitable for distributed computing even though it is actually implemented sequentially currently. The use of distributed computing with the proposed algorithm will improve the computational time.

In this research, design optimization by synthesizing the design alternatives in an unstructured design domain was investigated. The future improvement of the design

grammar and the computing perform will allow the optimization of trusses in an unstructured design domain for practical use. In addition to the combination with other soft computing techniques, such as neural network, that enables learning, will contribute to the development of intelligent design systems in the future.

REFERENCES

AISC (2001). *Manual of Steel Construction: Load and resistance factor design*, American Institute of Steel Construction, Chicago, IL.

Cheng, F.Y. and Li, D. (1997). "Multiobjective optimization design with Pareto genetic algorithm." *Journal of Structural Engineering,* 123(9), 1252-1261.

Coello, C.A. and Christiansen, A.D. (2000). "Multiobjective optimization of trusses using genetic algorithms." *Computers and Structures*, 75, 647-660.

Coello, C.A. and Veldhuizen, D.A. (2001). *Evolutionary algorithms for solving multi-objective problem*, Kluwer Academic Publisher, New York.

Dasgupta, D. (1994). "Handling deceptive problems using the structured genetic algorithm." *Proceedings of the First IEEE Conference on Evolutionary Computation: IEEE World Congress on Computational Intelligence*, IEEE Service Center, Piscataway, NJ, 1, 807-813.

Deb, K. and Gulati, S. (2001). "Design of truss-structures for minimum weight using genetic algorithms." *Finite Elements in Analysis and Design*, 37, 447-465.

De Jong, K.A. (1975). *An analysis of the behavior of a class of genetic adaptive system*, Ph.D. dissertation, University of Michigan, Ann Arbor, MI.

Dorn, W.C., Gomory, R.E., and Greenberg, H.J. (1964). "Automatic design of optimal structures." *Journal of Mechanics*, 3, 25-52.

Gage, P.J., Kroo, I.M., and Sobieski, I.P. (1995). "Variable-complexity genetic algorithm for topological design." *AIAA Journal,* 33(11), 2212-2217.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, Reading, MA.

Goldberg, D.E., Korb, B., and Deb, K. (1989). "Messy genetic algorithms: motivation, analysis, and first results." *Complex Systems*, 3(5), 493-530.

Hajela, P. and Lee, E. (1995). "Genetic algorithm in truss topological optimization." *International Journal of Solids and Structures*, 32(22), 3341-3357.

Holland, J. (1975). *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI.

Jenkins, W.M. (1997). "On the application of natural algorithms to structural design optimization." *Engineering Structures,* 19(4), 302-308.

Jenkins, W.M. (2002). "A decimal-coded evolutionary algorithm for constrained optimization." *Computers and Structures*, 80, 471-480.

Krishnakumar, K. (1989). "Micro genetic algorithms for stationary and non-stationary function optimization." *SPIE Intelligent Control Adaptation System*, 1196, 289-296.

Raich, A.M. (1999). *An evolutionary based methodology for representing and evolving structural design solutions*, Ph.D. dissertation, Department of Civil and Environmental Engineering, University of Illinois, Urbana-Champaign, IL.

Raich, A.M. and Ghaboussi, J. (1998). "Evolving structural design solution using an implicit redundant genetic algorithm." *Structural Multidisciplinary Optimization*, 20, 222-231.

Rajan, S.D. (1995). "Sizing, shape, and topology design optimization of trusses using genetic algorithm." *Journal of Structural Engineering*, 121(10), 1480-1487.

Rajeev, S. and Krishnamoorthy, C.S. (1992). "Discrete optimization of structures using genetic algorithms." *Journal of Structural Engineering*, 118(5), 1233-1250.

Rajeev, S. and Krishnamoorthy, C.S. (1997). "Genetic algorithms-based methodologies for design optimization of trusses." *Journal of Structural Engineering*, 123(3), 350-358.

Roston, G.P. and Sturges, R.H. (1996). "Using the genetic design methodology for structure configuration." *Microcomputers in Civil Engineering*, 11, 175-183.

Ruy, W., Yang, Y., and Kim, G. (2001). "Topology design of truss structures in a multicriteria environment." *Computer Aided Civil and Infrastructure Engineering*, 16, 246-258.

Ryoo, J. and Hajela, P. (2004). "Handling variable string lengths in GA-based structural topology optimization." *Structural Multidisciplinary Optimization*, 26, 318-325.

Schaffer, J.D. (1985). "Multiple objective optimization with vector evaluated genetic algorithms." *Proceedings of the First International Conference on Genetic Algorithms,* Lawrence Erlbaum Associates, Hillsdale, NJ, 93-100.

Shrestha, S.M. and Ghaboussi, J. (1998). "Evolution of optimum structural shapes using genetic algorithm." *Journal of Structural Engineering*, 124(8), 1331-1338.

Yeh, I.C. (1999). "Hybrid genetic algorithm for optimization of truss structures." *Computer-Aided Civil and Infrastructure Engineering*, 14, 199-206.

Zadeh, L.A. (1965). "Fuzzy set." *Information and Control*, 8, 338-353.

Zitzler, E., Deb, K., and Thiele, L. (2000). "Comparison of multiobjective evolutionary algorithms: Empirical results." *Evolutionary Computation*, 8(2), 173-195.

Zitzler, E. and Thiele, L. (1999). "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach." *IEEE Transactions on Evolutionary Computation,* 3(4), 257-271.

VITA

Sangwook Paik was born in Seoul, Korea, on July 18, 1973. He studied at In-ha University in Inchon, Korea, where he received the degree of Bachelor of Science in civil engineering in 2000. He enrolled in the master's program in civil engineering at Texas A&M University, where he worked as a research assistant. He worked with Dr. Raich, and his research focused on heuristic optimization for truss structures. He graduated in August 2005 with a M.S. degree in civil engineering from Texas A&M University (College Station, Texas). He may be contacted through the following addresses:

Permanent Mailing Address:

Hyundai Morning Side-2, 102-704

Shinhyun-Li, Opo-Eub, Kyungki-Do

South Korea

E-mail: swpaik@tamu.edu